P. 1877/91

10 1991

informatyka

Komunikacja z użytkownikami
Analizatory wykonania programów
Łączenie sieci komputerowych

WYDAWNICTWO SIGMA NOT

Cena 12 000 zł

SUPRA SP. Z O.O.
ul. Trybunalska 54, 60-325 POZNAŃ
telefon: 674392, telefaks: 236498
teleks: 412966

AUTORYZOWANY DYSTRYBUTOR FIRMY MANNESMANN TALLY OFERUJE DRUKARKI HEAVY DUTY

MT 645
drukarka mozaikowo-wierszowa

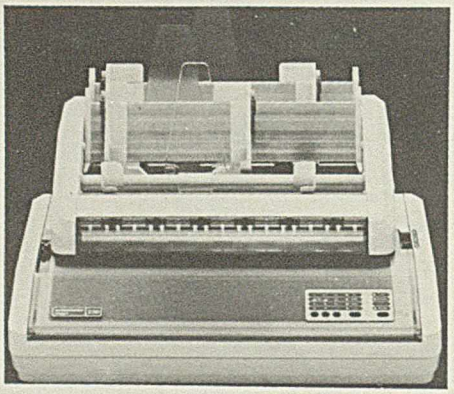
- doskonała jakość druku
- niezawodność 7000 h MTBF
- duża szybkość 450 wierszy/min.
- cicha praca
- bufor 27 KB
- liczba kopii 1 + 5
- pełna grafika 240 (dpi)
- polskie litery (opcja)
- wewnętrzny program testujący
- dwa interfejsy automatycznie przełączane
- podwójny system traktorów

MT 340
drukarka mozaikowa (18 igłowa)

- doskonała jakość druku
- niezawodność
- szybkość 400 zn/sek.
- cicha praca
- polskie litery (opcja)
- druk kolorowy (opcja)

MT 230
drukarka mozaikowa (9, 18, 24 igłowa)

- doskonała jakość druku
- niezawodność
- szybkość 300 zn/sek.
- cicha praca
- polskie litery (opcja)
- druk kolorowy (opcja)



MANNESMANN TALLY

PRAWDOPODOBNIE NAJLEPSZE DRUKARKI NA ŚWIECIE!

0/2/91

informatyka

Miesięcznik
ISSN 0542-9951
INDEKS 36124

P. 1877/91

nr 10 1991
Październik
Rok wydania XXVI

KOLEGIUM REDAKCYJNE:

mgr Jarosław DEMINET
mgr inż. Piotr FUGLEWICZ
dr inż. Wacław ISZKOWSKI
mgr Teresa JABŁOŃSKA
(sekretarz redakcji)
Władysław KLEPACZ
(redaktor naczelny)
dr inż. Wojciech MOKRZYCKI
mgr inż. Jan RYŻKO
dr Zdzisław SZYJEWSKI
mgr Hanna WŁODARSKA

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

WYDAWCA:
Wydawnictwo Czasopism i Książek
Technicznych SIGMA NOT
Spółka z o.o.
ul. Biała 4
00-950 WARSZAWA
skrytka pocztowa 1004

Redakcja:
01-552 Warszawa,
Pl. Inwalidów 10, p. 104, 105
tel. 39-14-34

Materiałów nie zamówionych
redakcja nie zwraca

W sprawach ogłoszeń prosimy zwracać się bezpośrednio do Redakcji lub Działu Reklamy i Marketingu 00-950 Warszawa ul. Biała 4 telefon: 20-31-24 telefaks: 20-31-16 teleks: 814550

W numerze:

Strona

Problemy komunikacji z użytkownikami i wewnątrz zespołu (1) — *Marian Kuraś* 2
Analizatory wykonania programów — *Tomasz Kalinowski* 10
Łączenie lokalnych sieci komputerowych — *Tomasz Kokowski, Marek Reformat, Maciej Stroiński* 17
Jak pisać programy satysfakcjonujące użytkownika? — *Andrzej Maciej Wierzba* 20
Procedury graficzne systemu RAMTEK RM-9460 — *Maria Bronowska* 25
Język specyfikacji SDL (2) — *Piotr Ostrowski, Marek Średniawa* 28

/unix

Studium klienta i usługowcy, czyli o Sun-owej koncepcji sieci otwartych (3). NFS od środka — *Jacek Surma* 34
Programowanie w języku awk (2) — *Władysław Majerski* 39

W najbliższych numerach:

- **Tomasz Kokowski, Marek Reformat i Maciej Stroiński charakteryzują konfigurowanie sprzętu i oprogramowania lokalnych sieci komputerowych.**
- **Andrzej P. Wojewoda opisuje lokalną sieć komputerową na przykładzie sieci IBM Token Ring.**
- **Krzysztof Chełpiński przedstawia, jak były tworzone przez firmę IBM ramy architektury dla komunikacji, od chwili wprowadzenia na rynek architektury SNA.**
- **Lech Pławecki i Katarzyna Stąpor omawiają doświadczenia z eksploatacji pakietu CASE, zakupionego przez Ośrodek Informatyki Śląskiego DOKP w Katowicach.**

Warunki prenumeraty

Przyjęcie prenumeraty – wyłącznie na podstawie dokonanej wpłaty na drukach dostarczanych dotychczasowym prenumeratorom przez Wydawnictwo, lub nowym – po uprzednim zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu Wydawnictwa.

Blankiet wpłaty – powinien zawierać następujące informacje: dokładna nazwa i adres (z kodem pocztowym) zamawiającego, tytuły zamawianych czasopism, ich liczbę i okres prenumeraty.

Wpłata – zgodnie z podanymi cenami należy dokonać w banku lub w UPT na konto podane na naszym blankiecie, tj:
Państwowy Bank Kredytowy III O/Warszawa nr: 370015-1573-139-11

Prenumeratorzy zbiorowi – osoby prawne obowiązują blankiety „Wpłata-Zamówienie". Cena normalna.

Prenumeratorzy indywidualni – osoby fizyczne obowiązują blankiety typu przekazy dla wpłat na rachunki bankowe. Cena normalna.

Prenumerata ulgowa – zgodnie z podaną ceną ulgową przysługuje wyłącznie osobom fizycznym, będącym członkami SNT, studentom i uczniom szkół zawodowych. Uczniowie szkół ogólnokształcących mogą zamówić w prenumeracie ulgowej tylko miesięcznik „Aura"
Uwaga! w podanym okresie prenumeraty można zamówić tylko po jednym egzemplarzu z każdego tytułu.

Prenumerata ze zleceniem wysyłki za granicę – cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa od ceny normalnej.
Należy podać dokładny adres odbiorcy za granicą.

Terminy przyjmowania prenumeraty:
– do 10 listopada na I, II, III, IV kwartał następnego roku
– do 28 lutego na II, III, IV kwartał br.
– do 31 maja na III i IV kwartał br.
– do 31 sierpnia na IV kwartał br.

Zmiany w prenumeracie, np. zmiana liczby tytułów, liczby egzemplarzy, rezygnacja z prenumeraty, można zgłaszać tylko w podanych terminach z mocą obowiązującą od następnego kwartału.

Egzemplarze archiwalne (z lat ubiegłych)
Można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa, ul. Mazowiecka 12 (tel. 26-80-16) lub zamówić pisemnie w Zakładzie Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek lub za zaliczeniem pocztowym.

Informacji o prenumeracie udziela: Zakład Kolportażu Wydawnictwa SIGMA-NOT Spółka z o.o., 00-716 Warszawa, ul. Bartycka 20, skr. 1004. Telefony: 40-00-21 wewn. 293, 295, 299 lub 40-30-86, 40-35-89.

Wstępna cena jednego egzemplarza na 1991 rok: normalna – 12 000 zł, ulgowa – 8400 zł

Wartość prenumeraty (w zł):
Normalna: kwartalna – 36 000, półroczna 72 000, roczna 144 000
Ulgowa: kwartalna – 25 200, półroczna 50 400, roczna 100 800

Uwaga: W przypadku zmiany cen w okresie objętym prenumeratą, prenumeratorzy zobowiązani są do dopłaty różnicy cen.

III Beskidzka Szkoła PTI
Szczyrk, czerwiec 1991

MARIAN KURAŚ
Zakład Zastosowań Informatyki
Kraków

Problemy komunikacji z użytkownikami i wewnątrz zespołu (1)

Znaczenie komunikacji w projektowaniu nie może być przecenione, lecz w naszej praktyce problematyka ta jest niedoceniana lub wręcz lekceważona[1]. Można tak uważać, jeśli weźmie się pod uwagę programy akademickiego kształcenia analityków systemów informacyjnych, studiów podyplomowanych oraz wszelkich kursów. W systemie kształcenia tematyka *public relations* była w ogóle nie zauważona i dotknęło to również dziedziny, w której umiejętności nawiązywania i utrzymywania kontaktów, pozyskiwania informacji i przekonywania mają podstawowe znaczenie.

Brak umiejętności komunikowania się w skali społecznej, powszechna nieświadomość ich znaczenia, jednoznaczna z nietworzeniem warunków ich rozwijania, a nawet lekceważenie tematyki zarówno przez ludzi uzdolnionych w tym kierunku, jak i osoby od urodzenia niekomunikatywne — wszystko to daje obraz ograniczeń i barier, na jakie muszą natrafić próby wprowadzania nowoczesnych metod zarządzania i ich wspomagania komputerowego. Próby dokształcania managerów i analityków organizacji lub analityków systemów informacyjnych mogą przynieść tylko ograniczone korzyści, gdyż nie odczuwa się powszechnie potrzeby komunikowania, nie docenia się jego efektów i nie widzi się potrzeby wymiany myśli w codziennym działaniu. Jest to efektem wieloletniego funkcjonowania scentralizowanych struktur, w których człowiek był traktowany jako jeden z zasobów produkcyjnych i mimo tworzonych pozorów oraz lansowania haseł o jego podmiotowości był traktowany przedmiotowo. W takim, ukształtowanym na modłę wojskową, systemie zarządzania nie istniała w zasadzie potrzeba społecznego komunikowania się. Jedynymi formami porozumiewania się były: polecenia i rozkazy do bezdyskusyjnego wykonania, instrukcje regulujące w sposób zunifikowany zachowania wszystkich wykonawców i sprawozdania przekazywane zwierzchnikom w sposób równie sformalizowany. Informacyjne więzi poziome (kooperacyjne) i ukośne (koordynacyjne) zostały praktycznie wyeliminowane i zastąpione więziami pionowymi. W strukturach formalnych w miejsce dobrowolnej wymiany informacji między członkami organizacji (np. w drodze negocjowania) pojawił się spór jako sposób dochodzenia racji z pozycji siły. W cenie była erystyka [21], podczas gdy techniki komunikowania się pozostawały nieznane dla szerokiej publiczności, a specjaliści nie znajdowali zainteresowania swą dziedziną. Obecnie komunikacja staje się przedmiotem zainteresowania osób pragnących podnieść swoje kwalifikacje i wzorem zachodnich systemów kształcenia jest umieszczana w programach różnych szkół businessu i kursów specjalistycznych.

Znaczenie komunikacji w projektowaniu systemów informacyjnych (SI)

Proces projektowania może być postrzegany i badany jako proces komunikowania się: wymiany informacji między użytkownikami[2] oraz między członkami zespołu projektującego. Ze znaczenia komunikacji zdaje sobie dobrze sprawę każdy analityk-projektant mający choć minimalny staż pracy. Błędy popełniane w toku procesu projektowania uświadamiają, jak ważna jest skuteczna komunikacja w zespole projektowym i jak wiele zależy od umiejętności nawiązywania, utrzymywania, a przede wszystkim wykorzystywania kontaktów z użytkownikami.

Znaczenie komunikacji w ramach zespołu projektowego jest doceniane od lat sześćdziesiątych. Problemom wynikającym z jej zakłóceń specjaliści starają się zaradzać jeszcze przed ich pojawieniem się — w przygotowaniu specjalistów i tworzeniu warsztatu projektanta[3]. W podręcznikach analizy — projektowania SI są omawiane wymagania dotyczące umiejętności analityka i jego warsztatu oraz przedstawiane rady i wskazówki praktyczne[4].

Celem rozwiązań przyjmowanych we współczesnych metodykach projektowania SI jest właśnie usprawnianie komunikacji. Zastosowane techniki i narzędzia mają zapewniać skuteczną komunikację w zespołach projektowych, a przede wszystkim komunikacji analityków z użytkownikami[5]. Wykorzystywane modelowanie SI jako metoda jego badania ma na celu osiągnięcie pełnego zrozumienia problemów użytkownika oraz ustalenia i zarejestrowania wszystkich istotnych do tworzenia SI cech i faktów. Aby to osiągnąć, konieczne staje się opracowywanie bardzo obszernej dokumentacji oraz rygorystyczne przestrzeganie procedur jej tworzenia, przechowywania i udostępniania. Zakres dokumentacji oraz pracochłonność jej tworzenia okazuje się w warunkach niedostatecznego wyposażenia w urządzenia biurowe (że pominie się tutaj będące standardem komputerowe wspomaganie metodyk tworzenia SI) przeszkodą nie

[1] Problematyka komunikowania się w zespołach projektowych była i jest oczywiście podejmowana w licznych pracach poświęconych problematyce projektowania [8, 15, 16, 22], przy czym jej ujęcie można określić jako normatywne lub badawcze. Jednocześnie brak jest opracowań mających charakter podręczników lub poradników dla adeptów zawodu projektanta (każdego zawodu), zawierających rady, zasady i wskazówki do rozwijania umiejętności skutecznego i efektywnego komunikowania się.

[2] Kategoria ta obejmuje różne grupy i role, w jakich użytkownicy występują wobec informatyków: klientów, kierownictwa i bezpośrednich użytkowników.

[3] Z badań czasu pracy kierowników wynika, że przeznaczają oni średnio 60% czasu na komunikację. W przeprowadzonym studium stwierdzono, iż jeden kierownik obserwowany przez okres 35 dni miał tylko 12 kwadransów, podczas których nie mówił. W innym studium ustalono, że kierownicy poświęcają na mówienie 80% swojego czasu pracy [6]. Często podkreślane w literaturze analogie między pracą kierownika i analityka SI nakazują przyjąć zbliżone wielkości dla analityka i na ich podstawie sądzić o znaczeniu komunikacji w jego pracy.

[4] Jako przykłady można wymienić liczne podręczniki zachodnie [1, 10, 19] lub starsze przetłumaczone na język polski [2, 4, 11,18]. W wymienionych podręcznikach znaczące fragmenty tekstu poświęca się zasadom współpracy między analitykami, a przede wszystkim z użytkownikami. W polskich podręcznikach eksponuje się tematykę technologiczną, a problematyka organizacji zespołu i komunikacji jest praktycznie pomijana.

[5] W opisach metodyk stworzonych po drugim kryzysie oprogramowania, jako cel ich stosowania wskazuje się umożliwienie sprawnej komunikacji między informatykami a użytkownikami oraz usprawnienie organizacji prac projektowych [1, 5, 7, 12, 14, 17, 20, 23].

do pokonania. W ten sposób efektywne metody tworzenia SI, stosowane z powodzeniem w krajach zachodnich, u nas albo nie mogą być stosowane, albo nie poprawiają w oczekiwanym stopniu komunikacji z użytkownikiem.

Przedstawione rozważania nie wyczerpują problematyki komunikacji, a w zasadzie pomijają najważniejsze jej aspekty. Nie uwzględnia się w niej najistotniejszych umiejętności — komunikacji interpersonalnej, bez znajomości której zastosowanie najlepszych narzędzi jest bardzo problematyczne, a jeśli nawet uda się je skutecznie wykorzystać do stworzenia potrzebnego użytkownikom SI, to nie można oczekiwać jego akceptacji. Należy bowiem pamiętać — a to często zarzucamy użytkownikom — iż nie są oni zwykle przekonani do samej komputeryzacji, w szczególności nie rozumieją nowego rozwiązania, nie umieją go ocenić, obawiają się jego negatywnego wpływu na ich sytuację jako pracowników, nie wiedzą sami, czego im potrzeba itd. Wobec takich ocen użytkowników powinny być doceniane zdolności i umiejętności motywowania, tłumaczenia, przekonywania, uzyskiwania informacji, instruowania itd.

Kontakty z użytkownikiem nie ograniczają się tylko do samego procesu projektowania i nie polegają wyłącznie na gromadzeniu danych projektowych i przedstawianiu użytkownikowi gotowego rozwiązania. Rozpoczynają się one z chwilą uzyskania pierwszej informacji o zainteresowaniu **użytkownika-klienta** (tego, który zamawia usługi, a później płaci), są kontynuowane z **użytkownikiem-kierownikiem** (który ustala cele i ograniczenia projektu (oraz z **użytkownikiem końcowym**), który ma potrzebną informatykowi wiedzę o funkcjonowaniu organizacji i określone potrzeby wspomagania) oraz z wszystkimi wymienionymi grupami przy zakończeniu projektu i w toku eksploatacji. Zależność przebiegu projektu od skuteczności komunikacji między informatykami a wszystkimi grupami użytkowników jest bezsporna i zwłaszcza informatykom są nieodzowne umiejętności nawiązywania i utrzymywania kontaktów, negocjowania, przekonywania, tłumaczenia, raportowania, instruowania itd. Są one niezbędne ze względu na postawy użytkowników oraz ich ograniczone zdolności komunikowania się.

Komunikacji interpersonalnej trzeba więc przyznać pierwszeństwo przed formalnymi metodami i technikami organizacji i porozumiewania się w zespole projektowym. Ten drugi obszar jest w większym stopniu określony i rozpracowany, a także możliwy do przyswojenia, pod warunkiem jednak stworzenia odpowiednich warunków — zrozumienia i współpracy oraz świadomości wspólnoty interesów użytkowników i informatyków. Warunki te można kształtować, jeśli właśnie informatycy będą zdolni do nawiązywania i utrzymywania kontaktów, stworzenia odpowiedniej atmosfery porozumienia, jeśli nasza umiejętność słuchania pozwoli na zrozumienie sytuacji i problemów użytkowników, a komunikatywność na przekazywanie naszych pomysłów.

Proces komunikacji

Przez komunikację powszechnie rozumie się proces wysyłania i otrzymywania wiadomości (komunikatów) [3]. Komunikacja odbywa się werbalnie (komunikaty mówione i pisane) i niewerbalnie. Z pierwszą formą komunikacji wiąże się pięć czynności:

- **mówienie**, będące podstawowym sposobem przekazywania komunikatów; w ten sposób przekazuje się większość komunikatów, przy czym sposoby ich przekazywania są bardzo zróżnicowane (narady, spotkania, negocjacje, wywiady, instruktaże, rozmowy telefoniczne itp.);
- **pisanie**, stawiające większe wymagania niż mówienie, ze względu na czasochłonność oraz konieczność jasnego sformułowania komunikatu i brak możliwości natychmiastowego wyjaśnienia jego szczegółów[6]; formy komunikacji pisemnej są niewiele mniej zróżnicowane niż ustnej i obejmują: listy, raporty, notatki, materiały reklamowe itp.;
- **słuchanie**, lekceważone przez większość ludzi; polega na słuchaniu, rozumieniu i interpretowaniu komunikatów; szczególnie rzadka i pożądana jest umiejętność słuchania empatycznego (empatia — wczucie się) bez konieczności angażowania się emocjonalnego, pozwalające lepiej zrozumieć interlokutora;
- **czytanie**, zaniedbywane przez wielu specjalistów[7], wymaga — jeśli ma być skutecznie wykorzystanie — specjalnych umiejętności fotograficznego i szybkiego przeglądania oraz selekcji tekstów, co osiąga się w wyniku konsekwentnego treningu;
- **myślenie**, polegające na operowaniu symbolami, którym indywidualnie nadaje się odpowiedniki słowne, co oznacza, iż słyszy się to, co jest się w stanie (chce się) usłyszeć.

Niewerbalne formy stanowią ponad 65% komunikacji, górując nad formami werbalnymi. Oznacza to, że komunikaty niewerbalne mogą modyfikować zamierzony odbiór wiadomości werbalnych — wzmacniać je, uzupełniać, zastępować, potwierdzać bądź osłabiać lub zaprzeczać. Jako podstawowe formy komunikacji niewerbalnej należy wymienić:

- **język ciała** (ang. *body language*), czyli fizyczne zachowania, takie jak sposób chodzenia, siedzenia, gestykulacja, grymasy, mimika i inne znaki;
- **aparycja**, niedoceniana bądź wręcz lekceważona przez jednych, podczas gdy inni przydają jej zbyt wielkie znaczenie;
- **instynkt terytorialny** (ang. *distance*), który jeśli jest naruszany w określonym czasie wobec osoby, może być czynnikiem niesprzyjającym, a w innych przypadkach wejście w czyjeś „terytorium" jest odczytywane jako gest przyjazny i sprzyja kontaktom;
- **dystans** (ang. *touching behaviour*) — dotyk jest bardzo ważną formą komunikacji, częściej niedocenianą niż przecenianą; zróżnicowanie jej odbioru (uścisk dłoni, poklepanie po plecach, oparcie się, objęcie itp.) powoduje, że rezygnuje się ze zmniejszania dystansu wobec ryzyka negatywnego odbioru, co wskazuje na potrzebę uczenia się „zachowań dotykowych", tzn. przemyślenia i eksperymentowania przed ich stosowaniem w praktyce;
- **parajęzyk**, czyli żargon profesjonalny oraz wtręty językowe pozwalające na chwilę zastanowienia (np.: „eee", „yyy", „prawda"), które z jednej strony czynią język żywym i ułatwiają porozumienie, ale jeśli są nadużywane (żargon!) zakłócają słuchaczowi odbiór przekazywanego komunikatu i mogą nastawiać negatywnie do przemawiającego;
- **rekwizyty** (ang. *artifacts*) — ubranie i przybrania (biżuteria, kosmetyki i inne atrybuty, którymi chcemy podkreślić przynależność do..., upodobnić się do..., zaświadczyć, że..., itd.);
- **otoczenie**, w którym podejmuje się rozmowę, ma oddziaływać na postawę i nastawienie interlokutora i może sprzyjać lub przeszkadzać w komunikacji;
- **czas**; sposób wykorzystywania swojego czasu i czasu innych wiele mówi o osobie: szanowanie i oszczędne gospodarowanie czasem budzi zaufanie, podczas gdy spóźnianie się i ciągły pośpiech rodzą rezerwę;
- **niewerbalne komunikaty pisane** mogą zawierać dodatkowe informacje o nadawcy i jego nastawieniu w postaci np. formy listu, sposobu jego zredagowania, użytego papieru itp.

---

[6] Umiejętność komunikowania się w postaci pisanej jest istotna zwłaszcza w pracy analityka, który pełni funkcję pośrednika między użytkownikami a technologami i z tej przyczyny jest zmuszony do dokumentowania wyników swej pracy, formalnego rejestrowania ustaleń, zasięgania ekspertyz i opinii przygotowywania instrukcji dla użytkowników, raportowania itp.

[7] Jakże często słyszymy skargi w rodzaju: *nie mam czasu na przeczytania czasopisma fachowego, nie nadążam z czytaniem korespondencji* itp. Te same osoby muszą później stracić dużo więcej czasu na uzyskanie informacji, jakie można uzyskać systematycznie czytając (że pominiemy tu konieczność poszerzania swojej wiedzy).

Komunikacja w organizacji służy rozwiązywaniu problemów gospodarczych[8]. Problemy te można zaklasyfikować do jednej z trzech grup:

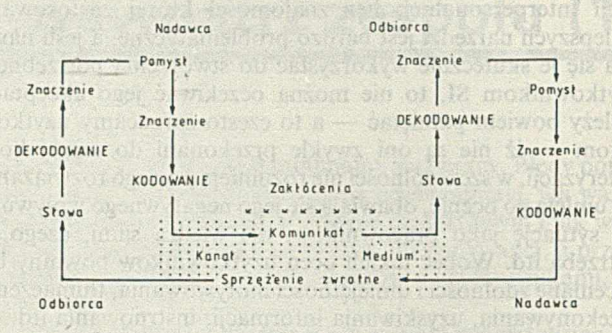
- problemy związane z zadaniami wykonywanymi przez organizację:
  - — ustalanie celów i polityki działania;
  - — osiąganie ustalonych celów i realizowanie polityki;
  - — pomiar i ocena rezultatów;
- problemy związane z ludźmi:
  - — zatrudnianie i wprowadzanie nowych pracowników;
  - — motywowanie i tworzenie korzystnej atmosfery;
  - — ocena wyników pracy;
- inne problemy:
  - — kontakty z kontrahentami (klientami i dostawcami);
  - — kontakty z konkurencją;
  - — współpraca z instytucjami publicznymi: administracją, władzami terenowymi, związkami zawodowymi itp.

W procesie komunikacji zarówno nadawca, jak i odbiorca podlegają wpływom różnorodnych czynników, z których najważniejszymi są: **język, system społeczny, religia, postawy, wykształcenie, technika, polityka i system prawny**. Razem stanowią one pewien poziom kultury (cywilizacyjny), który determinuje sposób komunikowania się, a ten odzwierciedla nasz poziom kulturalny.

Wpływ tych czynników na nadawcę komunikatu można zreasumować w następujących punktach:

- **język** dostarcza wzorców do klasyfikowania własnych doświadczeń i obserwacji;
- **system społeczny** stwarza pewne typowe więzi (rodzina, koledzy, przełożeni itp.), które kształtują modele komunikowania się;
- **religia** wielu ludziom daje podstawy egzystencjonalne i wzorce, które określają sposoby traktowania otaczającego świata oraz innych ludzi;
- **postawy**, jakie wykształcimy w sobie wobec zjawisk i obiektów otoczenia (czasu, wiedzy, pracy, korzyści materialnych itp.), wpływają na kontakty i sposoby komunikowania się ludzi;
- **wykształcenie**, im wyższe, tym zapewnia większe potencjonalne zdolności komunikowania się, system edukacyjny powinien rozwijać umiejętność komunikowania oraz promować akceptowane wzorce;
- **technika** może poważnie zwiększać możliwości komunikowania się ułatwiając, przyspieszając i usprawniając oraz zwiększając możliwości nadawania i przyjmowania komunikatów;
- **system polityczny** stwarza pewne ogólne ramy komunikacji; ograniczenia wolności słowa, system biurokratyczny i zakres samorządności wpływa na możliwości formułowania sądów i opinii, a w konsekwencji na cały proces komunikowania;
- **system prawny** wyznacza obszar legalności wszelkich działań, wyznaczając w ten sposób cenę ogłaszania sądów i opinii niezgodnych z rzeczywistością lub naruszających czyjeś dobro, przy czym zasadnicze znaczenie ma stopień respektowania reguł prawnych.

Te same czynniki oddziałują na odbiorcę komunikatu, przy czym jego poziom kultury i wykorzystywane wzorce są unikalne, odmienne od nadawcy. O tym musi się pamiętać opracowując strategię komunikacji. Odmienność poszczególnych grup uczestniczących w procesie tworzenia SI jest postrzegana, jednak nie popełni się nadużycia, jeśli stwierdzi się, że nie jest ta odmienność uwzględniana w kontaktach między informatykami a poszczególnymi grupami użytkowników[9].



Rys. 1. Proces komunikacji (na podstawie [3])

Na rysunku 1 przedstawiono schematycznie przebieg procesu komunikacji. Proces ten odbywa się w zamkniętej pętli, której punktem początkowym jest pomysł, powstający w rezultacie percepcji pewnego komunikatu lub zdarzenia w otoczeniu (rozwiązanie problemu, pomysł rozwiązania organizacyjnego itp.). Pomysł wymaga jego przedstawienia na przykład współpracownikom, jednak zanim to nastąpi, konieczne jest uświadomienie celu, jakiemu ma on służyć. Jest to nieodzowne do sformułowania strategii komunikacji[10], a także do określenia najlepszego sposobu przesłania. Wybór sposobu przekazania komunikatu — ustnego lub w formie pisanej — odpowiedniego do treści i innych warunków nie jest obojętny, gdyż ma zapewnić osiągnięcie ustalonego celu. Odmienny skutek osiąga się, jeśli tę samą wiadomość przekazuje się ustnie (bezpośrednio lub np. telefonicznie) i pisemnie (list, standardowy druk, teleks itp.).

Kodowanie wiadomości jest kolejnym krokiem, w którym stosownie do wybranego ośrodka przekazywania komunikat jest formułowany werbalnie. Trzeba przy tym brać pod uwagę to, że słowa mają oddawać pewne znaczenia, lecz ich nie zawierają, jest ono nadawane słowem przez odbiorcę. Dlatego dobór odpowiedniego słownictwa, formy i sposobu ich prezentacji ma kluczowe znaczenie dla zrozumienia. Dlatego trzeba przestrzegać co najmniej kilku podstawowych zasad.

- **Bądź precyzyjny**, unikaj terminów i sformułowań abstrakcyjnych:

zamiast powiedzieć:	**powiedz:**
Korzyści są wątpliwe.	Korzyści w ogóle nie szacowano
X spóźnia się do pracy.	X przychodzi codziennie o 9.40.
Y jest najlepszym analitykiem	Y wdrożył na czas trzeci projekt w tym roku
Projekt przyniesie korzyści wymierne i niewymierne.	Wdrożenie projektu zwiększy sprzedaż o min. 15% i ... .

- **Bądź ukierunkowany na użytkownika**, używaj sformułowań odpowiednich do jego wieku, przygotowania ogólnego i specjalistycznego:

do kogoś starszego:	**do młodszej osoby:**
Pan jest bardzo dobrze poinformowany.	Ty świetnie wiesz, co jest grane.
do osoby wykształconej:	**do osoby bez wykształcenia:**
On ma zmysł polityczny.	On ma nosa do ludzi.

[8] Problemy te pojawiają się w procesie tworzenia SI jako zadania do rozwiązania przez informatyków na drodze stworzenia odpowiednich narzędzi wspomagających komunikację — ustalania jej strategii i utrzymywanie kontaktów. Ponadto problemy komunikacji o podobnym charakterze występują w działalności informatyków wykonujących swoje zadania na rzecz użytkowników.

[9] W szczególnie złej sytuacji znajdują się zwykle użytkownicy końcowi, których wiedza o obiekcie jest analitykom najbardziej potrzebna i którzy najbardziej potrzebują wyjaśnienia celów zmian i warunków ich wprowadzania. Są oni traktowani często jako nieznaczący i nieliczący się przez wszystkich pozostałych uczestników, a czasami w taki sposób odbierają fakt decydowania bez ich udziału o sprawach dla nich istotnych.

[10] Do wyboru pozostają dwie podstawowe strategie: informowania (ang. *tell strategy*) lub motywowania (ang. *sell strategy*).

do specjalisty:	do niespecjalisty:
Trzeba zastosować podejście partycypacyjne.	Trzeba zaangażować użytkowników do tworzenia systemu informacyjnego, żeby zyskać ich akceptację.

● **Działaj odpowiednio do sytuacji**, formułuj propozycje odpowiednio do sytuacji:

zalecaj, polecaj:	sprzedaj:
Przyjdzie czas, że będziecie rozważać potrzebę wykorzystania takiego oprogramowania.	Proszę odwiedzić naszą firmę. Pokażemy Panu oprogramowanie, które przypuszczam może Pana zainteresować.

i właściwie używaj słów określających działanie:

przed:	po:
Pracował w dwóch zespołach.	Opracował dwa projekty.
Pracował w charakterze kierownika zespołu.	Kierował zespołem 12 analityków.
Brał udział we wdrażaniu dwóch systemów.	Wdrożył dwa systemy.

● **Bądź dobrze zorganizowany**; formułując komunikat podkreśl jego znaczenie umieszczając kluczowe informacje na początku lub na końcu paragrafu, wywód przeprowadź w sposób logiczny, spójny i zwięzły, używaj słów i zwrotów łączących zdania itd.;

● **Bądź sobą**; język, którego używamy, musi być zgodny z osobowością i stylem, w przeciwnym razie przesłana wiadomość może nie być odebrana zgodnie z naszymi intencjami.

Zakodowany komunikat jest następnie wysyłany do odbiorcy, co nie kończy zwykle pracy nad pomysłem. Odbiorca dekoduje wiadomość, interpretując i nadając znaczenie użytym słowom i formie, przy czym wynik odbioru jest uzależniony od zrozumienia kontekstu, wykorzystanego słownictwa, doświadczenia i zdolności do przezwyciężania barier i zakłóceń komunikacji. Udzielenie odpowiedzi, jak również jej brak, jest **sprzężeniem zwrotnym**, po którym role nadawcy i odbiorcy odwracają się. Należy w tym miejscu zwrócić uwagę na znaczenie sprzężenia zwrotnego w pracy analityka. Zasadnicza część komunikatów przekazywanych przez niego do użytkowników jest obliczona na otrzymanie odpowiedzi zawierającej informacje o wykonywanej pracy, opinie, sugestie, potrzeby, wymagania, oceny itd.

Omówienie procesu komunikacji należy uzupełnić uwagami na temat **ograniczeń i zakłóceń**. Zakres zakłóceń jest bardzo zróżnicowany i obejmuje zakłócenia akustyczne (hałas, zła dykcja mówiącego), wizualne (niewyraźne pismo, słaby druk, wada wzroku), zapachowe (nieprzyjemny zapach w otoczeniu mówiącego), dotykowe (poklepywanie interlokutora, ściskanie ręki), dotyczące postaw (uprzedzenia rasowe, niechęć do wąsatych i brodatych, osób oficjalnych, pewnych opcji politycznych itp.). Należy zawsze pamiętać, że wiadomość przekazana nigdy nie jest jednoznaczna z otrzymaną. Jest to spowodowane również zakłóceniami, których źródłami są: przygotowanie ogólne i zawodowe, zdolność percepcji, umiejętność porozumienia się, zachowania itd.

Strategie komunikacji

Zamierzenia, ich cele i sytuacje w toku prac nad tworzeniem SI wymagają opracowania **strategii komunikacji**[11], czyli zaplanowanie wszystkich szczegółów i aspektów kontaktów z użytkownikami. Planowanie komunikacji wymaga rozwiązania przede wszystkimi następujących kwestii:
— jasnego i jednoznacznego określenia celów działania, co ułatwia znalezienie odpowiednich rozwiązań komunikacji — jej adresatów, zawartości, struktury, tonu itd.;
— rozważenia wpływu komunikacji na organizację, współpracowników, przełożonych, użytkowników, przebieg prac nad projektem itp.:
— redukowania bądź zapobiegania powstawaniu postaw obronnych użytkowników, ich obaw i niepewności, które mogą zasadniczo zakłócić przebieg komunikacji;
— zapewnienia wiarygodności nadawcy i stworzenia przesłanek akceptacji przez odbiorcę.

Po sprecyzowaniu merytorycznego celu komunikacji należy wybrać jednę z trzech podstawowych strategii:

● informowania (także polecania, zalecania, rozkazywania, przekazywania), przy której podstawowym problemem jest uzyskanie pełnego zrozumienia przekazywanej wiadomości przez odbiorcę;

● motywowania (sprzedawania), która ma w celu przekonanie do komunikowanych idei lub do podjęcia sugerowanego sposobu działania;

● rozwiązywania (wydobywania — ang. *resolve strategy*), z którą mamy do czynienia wówczas, gdy nadawca spodziewa się usłyszeć w odpowiedzi coś niemiłego (np. w przypadku wypominania błędnie lub niekompletnie przedstawionych potrzeb, zwracania uwagi projektantowi, który niepoprawnie czy błędnie opracował dokumentację itp.).

Podstawowymi cechami, na jakie trzeba zwracać uwagę przy stosowaniu strategii informowania są:
— trafność i precyzja przekazu: poprawność, aktualność, wiarygodność źródeł, zgodność z opinią odbiorcy, precyzja sformułowania itp.;
— kompletność: zakres przekazu, nadmiar lub niedobór informacji, znajomość treści przekazu przez odbiorcę, zgodność z jego zainteresowaniami, obowiązkami itp.;
— klarowność: język przekazu, słownictwo, sposób sformułowania, złożoność zdań, logiczna spójność wywodu, użycie przykładów, ilustracji itp.

Strategia motywowania ma charakter perswazji, przekonywania i dlatego jest szczególnie przydatna dla analityka-projektanta SI. Większość zadań, jakie ma do wykonania, powinna mieć na celu sprzedanie własnych pomysłów, zachęcanie do udziału w wykonywanych pracach, prezentację własnych przemyśleń, zachęcanie do posługiwania się oferowanymi narzędziami itp. Przekaz, jaki nadawca przedstawia odbiorcy przy zastosowaniu tej strategii, powinien uwzględniać następujące czynniki:

● motywacje odbiorcy, każdy kieruje się w większym stopniu własnymi potrzebami, przekonaniami, wyobrażeniami lub uprzedzeniami niż logiką i siłą motywacji;

● wiarygodność nadawcy; zaufanie do nadawcy, uznanie jego odpowiedzialności i ocena kompetencji jest podstawą akceptowania[12] przekazywanych komunikatów;

● spójność, logiczność przekazu, przedstawienie wymownych danych lub trafnych przykładów współbrzmiących z prezentowanymi wnioskami i opiniami pozwala uzyskać poparcie dla przedstawianych pomysłów.

[11] Określenie: *strategia komunikacji* zostało bezpośrednio zapożyczone z języka angielskiego, ale ze względu na znaczenie oraz złożoność ujawniających się problemów wydaje się celowe traktowanie problemów komunikacji z użytkownikami jako kluczowych dla powodzenia wszelkich innych działań. Z tego powodu przyjęto nieco nadęte sformułowanie: *strategia komunikacji*.

[12] Przyjmowania jako prawdziwe, zgodne ze stanem faktycznym oraz akceptowane jako poprawne, nieodzowne lub korzystne. W literaturze przedstawia się liczne przykłady — pozytywne i negatywne — znaczenia wiarygodności jako podstawy skuteczności przekazu.

Strategia rozwiązywania — jak się uważa — ma coraz większe znaczenie w komunikacji między przełożonymi a podwładnymi. Umiejętność radzenia sobie w trudnych, nieprzyjemnych okolicznościach umożliwia rozwiązywanie problemów i wychodzenie z trudnych sytuacji. Ponadto umiejętność uzyskiwania i wykorzystania nieprzyjemnych opinii pozwala znaleźć i usuwać przyczyny niekorzystnych zjawisk w organizacji. Strategię tę można stosować przestrzegając:

- stosowania zasad wynikających z teorii stosunków międzyludzkich (ang. *human relations theory*), czyli zdolność rozpoznawania oraz okazania wrażliwości na cudze problemy;
- tworzenia odpowiedniej atmosfery; ludzie są zdolni do zaakceptowania nawet najgorszych wiadomości, jeśli są one przekazywane w sprzyjającej atmosferze i odpowiednim tonem — pisma, okólniki, oficjalne komunikaty itp. nie tworzą korzystnych warunków do komunikowania złych wiadomości;
- wybrania odpowiedniego momentu przekazania kluczowej wiadomości w toku rozmowy; zbyt wczesne podanie złej wiadomości może spowodować, że cały przekaz zostanie przyjęty negatywnie i odrzucony, a pozostałe istotne wiadomości będą zignorowane.

W praktyce stosuje się strategie mieszane, bowiem tego wymagają złożone cele i okoliczności, w jakich odbywa się komunikacja. Osoba odpowiedzialna za wdrożenie projektu przedstawia nowe rozwiązania SI (informowanie), starając się wzbudzić zainteresowanie i akceptację pomysłu (motywowanie). Może przy tym napotkać opór personelu, z czym będzie musiała sobie poradzić, odpowiednio prezentując istotne elementy wiadomości (rozwiązywanie). W takiej typowej sytuacji trzeba zastosować wszystkie rozważane strategie, a właściwie odpowiednio je zintegrować. Uzyskanie zgody na zmianę terminu ukończenia projektu w pierwszym rzędzie będzie wymagało przedstawienia faktów (informowania). Następnie należy przedstawić korzyści, jakie zostaną osiągnięte i mankamenty, które wyeliminuje się, jeśli czas opracowania projektu będzie wydłużony (motywowanie). Po takim przygotowaniu można przystąpić do zakomunikowania potrzeby zmiany terminu (rozwiązanie).

Niezależnie od strategii komunikacji, na osiągnięcie jej efektów końcowych ma wpływ sprzyjająca atmosfera, pozytywne nastawienie i dobra wola jej uczestników. Dlatego podstawowym zadaniem inicjatora komunikacji (a także jej pozostałych uczestników) jest stworzenie odpowiedniej atmosfery, a przede wszystkim pozytywnego nastawienia i wzajemnej życzliwości. Dobra wola uczestników oznacza ich ocenę wiarygodności, konsekwencji, uczciwości oraz sympatię dla nadawcy. Nadawca powinien zatem zabiegać o takie pozytywne oceny swojej osoby pamiętając, że są one podstawą zaufania do niego i do firmy, którą reprezentuje. Istnieje wiele możliwości pozytywnego nastawiania odbiorców komunikatów, ale zawsze należy przestrzegać następujących zasad [3, 6]:

● Bądź dyplomatą

Mówienie bezpośrednie, bezpośredni sposób bycia niekoniecznie musi oznaczać brak taktu lub wręcz arogancję. Każdy z nas oczekuje od innych zrozumienia dla siebie, swojej sytuacji, potrzeb, możliwości itd. Zrozumienie potrzeb innych, wczucie się w ich sytuację zawsze pomaga w stworzeniu dobrej atmosfery i ułatwia porozumienie. Bezpośrednie przedstawianie swych podejrzeń i przypuszczeń w każdym przypadku powoduje, że odbiorca czuje się zagrożony i w rozmowie będzie bronił się, a więc nie będzie skłonny do współpracy. W podanych poniżej przykładach wypowiedzi w tych samych sytuacjach wyraźnie widać różnicę i można domyślać się reakcji uczestników rozmowy.

nie mów nigdy:	**raczej powiedz:**
Oczywiście nie czytałeś mojego listu, w którym ...	Jak sugerowałem w moim liście z dnia ...
Będzie Pan żałował swojej decyzji	Jestem przekonany, że przemyślał Pan wszystkie aspekty swojej decyzji.
Wasze sugestie znowu są niezgodne z naszą umową.	Proponujemy oprzeć wspólne rozstrzygnięcie zgodnie z umową.
Znowu spóźniłeś się do pracy pół godziny.	Rozumiem, że od początku tygodnia wychodzisz z pracy bardzo późno.

● Bądź bezpośredni, traktuj innych prywatnie, indywidualnie.

Każdy lubi być traktowany indywidualnie, prywatnie — skraca to dystans i ułatwia porozumienie[13]. Warto porównać poniższe warianty zwracania się do rozmówcy i ocenić, przy których ja, jako odbiorca, będę lepiej traktowany i które pozytywnie nastawią mnie.

oficjalnie:	**bezpośrednio:**
Zmiany, jakie wprowadza się do systemu informacyjnego pozwolą na zwiększenie skuteczności i efektywności	Sądzę, że zaproponowane zmiany ułatwią pracę twoją, Piotrze, jak też twoich współpracowników.
Zawiadamia się wszystkich pracowników, że od dnia 91.03.28 likwiduje się	Szanowni Państwo; W związku z sytuacją naszego przedsiębiorstwa jesteśmy zmuszeni do rezygnacji z ... .

● Staraj się pomóc rozmówcy

Wszelkie kontakty profesjonalne wiążą się z pewnym stresem, z czym spotykamy się przy rozmowach informatyków z użytkownikami. Stres ten jest widoczny w zachowaniach użytkowników, którzy zwykle dostrzegają w podejmowanej komputeryzacji zagrożenia, a ponadto mają nieuzasadnione kompleksy wobec analityków. Skutkiem tego jest przyjmowanie przez nich postaw defensywnych; zadaniem analityka jest stworzenie atmosfery współpracy. Znajomość czynników powodujących ujawnianie się postaw: defensywnych i współpracy rozmówców jest nieodzowna do ich kształtowania. Czynniki przyczyniające się do ujawniania się postaw zestawiono w tabeli 1.

Tabela 1. Czynniki kształtowania postaw: defensywnych i współpracy (na podstawie [9])

Postawy defensywne:	
1. Ocena	— osądzanie innych ludzi, ich postaw i zachowań
2.Sterowanie	— decydowanie za innych, forsowanie gotowych rozwiązań
3. Strategia	— manipulacja, niejasne, niejednoznaczne motywy
4. Neutralność	— demonstrowanie braku zainteresowania, lekceważenie interesów innych ludzi
5. Wyższość	— podkreślanie przewagi nad innymi, postawa *jestem-lepszym-od-ciebie*
6. Zadufanie	— demonstrowanie postawy *ja-wiem-wszystko*
Postawy współpracy:	
1. Prezentacja	— obiektywny opis sytuacji lub oceny
2. Podejście merytoryczne	— orientacja problemowa, przyjęcie za cel rozwiązania problemu, rezygnacja z przyjmowania z góry rozwiązań
3. Spontaniczność	— otwartość, eliminowanie premedytacji
4. Empatia	— wczuwanie się w cudze położenie, demonstrowanie wrażliwości na cudze potrzeby
5. Równość	— gotowość do współuczestniczenia w działaniu przy założeniu wzajemnego zaufania i poszanowania
6. Poświęcenie	— gotowość do elastycznego postępowania i eksperymentowania ze swymi zachowaniami, postawami i pomysłami

● Bądź uczciwy

Wszelkie komunikaty dla odbiorców niosą informację o uczciwości samego pracownika, a także firmy (zespołu), który reprezentuje. Obietnice, półprawdy, kłamstwa przedstawiane

[13] Trzeba oczywiście brać pod uwagę pewne standardy i konwenanse. Nadmiernie bezpośrednie traktowanie współpracowników może odnieść efekt odwrotny od zamierzonego. Nie ułatwiają zadania formy grzecznościowe języka polskiego. Dominujące formy zwracania się do osób nie będących bliskimi znajomymi (pan, pani) nie za bardzo współbrzmią z nazwiskiem, a z imieniem są odbierane jako mało naturalne, czasami jako zbyt poufałe. Przy ograniczeniach zmiany formy zwracania się na *ty* traci się możliwość skracania dystansu przez używanie w rozmowie imienia lub nazwiska.

np. użytkownikowi mogą przynieść pewne korzyści doraźne, jednak straty, jakie poniesie się na skutek utraty wiarygodności, mogą się okazać bez porównania większe. Decyzja co do zakresu informacji przekazywanej w komunikatach zależy od nadawcy, a jego etyka[14] dyktuje, co może i powinno być powiedziane, a czego nie należy przekazywać odbiorcom.

● **Szukaj i używaj pozytywnych argumentów i sformułowań**

Elementarną zasadą pozytywnego nastawiania odbiorcy komunikatu jest eliminowanie negatywnych sformułowań i używanie tylko argumentów pozytywnych (tabela 2). Skupienie się na pozytywach zawsze eliminuje elementy oceny i obawy związane z negatywnymi lub niepewnymi konsekwencjami działania.

Tabela 2. Przykładowy zestaw słów o znaczeniu negatywnym i pozytywnym

W liście zestawiono słowa i zwroty o wyraźnymy odbiorze negatywnym lub pozytywnym powszechnie spotykane w praktyce. Lista jest oczywiście otwarta i należy ją uzupełniać.

Negatywne	Pozytywne
problem	pytanie
ryzyko, niepewność	bezpieczeństwo
szach (potrzask)	szansa
niepowodzenie	sukces
strata	korzyść, zysk
obawa	okazja
sprzedać	uzyskać korzyść
przedstawiciel	doświadczenie
trochę	w pełni
być może, jeśli	napewno
opóźnienie	zmiana terminu
poszukiwania	promocja
przypuszczalnie	informacja
'wy'	'my', Pan, Pani
ewentualnie	z pewnością
zmuszać	powodować
nie będziemy mogli ...	możemy tylko
oddalamy ...	jedyne, co możemy ...
żałować	
przecież	
nigdy	
niestety	
szkoda	
z przykrością	
przymus, jesteśmy zmuszeni	

Używanie pozytywnych sformułowań i argumentów nastawia pozytywnie i optymistycznie rozmówcę. Poniżej przedstawiono odmienne sposoby sformułowań dotyczące tych samych sytuacji.

nastawienie pozytywne:

W związku z niedotrzymaniem terminu przekazania ... informujemy z przykrością, że nie możemy dotrzymać ustalonego wcześniej terminu.

Preferencyjnych akcji prywatyzowanego przedsiębiorstwa nie mogą zakupić pracownicy, którzy przepracowali mniej niż trzy lata.

Pragniemy poinformować zainteresowanych, że nie będą przyjmowane nieudokumentowane reklamacje dotyczące funkcjonowania systemu.

nastawienie negatywne:

Prosimy o możliwie szybkie przekazanie ... . Jeżeli otrzymamy ... do ... będziemy mogli przekazać zamówione opracowanie przed ... .

Preferencyjne akcje prywatyzowanego przedsiębiorstwa mogą kupić pracownicy, którzy przepracowali co najmniej trzy lata.

Prosimy o przekazywanie wszelkich uwag, sugestii i reklamacji do ... . Ich szybkie wykorzystanie będzie możliwe, jeżeli dostaniemy dokumentację wykrytych niesprawności.

Kontakty podejmujemy, kiedy trzeba rozwiązać problem, przy czym o jego istnieniu jest przekonany inicjator rozmów. Musi on odpowiednio przygotować się do rozmowy z użytkownikiem, jeśli chce zapewnić jego zaangażowanie. W tym celu w pierwszym rzędzie należy odpowiedzieć sobie na pytanie: kim jest rozmówca? Od odpowiedzi na postawione pytanie zależy sposób prezentacji wiadomości, prowadzenia rozmowy itd.

Drugie pytanie, na jakie należy udzielić odpowiedzi przygotowując się do kontaktów, brzmi: **jakie są potrzeby i motywacje rozmówcy?**

Znajomość potrzeb rozmówcy i jego psychologicznych motywacji pozwala na przygotowanie i wykorzystanie rozmów do osiągnięcia założonych celów. Teorie zachowań A. Maslowa, F. Herzberga i McGregora[15] dostarczają wiedzy o hierarchii potrzeb pozwalającej na unikanie błędów w komunikowaniu się. Teorie te nie dają jednak wskazówek, w jaki sposób wykorzystać tę wiedzę do poprawienia komunikacji. W większym stopniu jest tu przydatna **teoria świadomości zależności** (ang. *relationship awareness theory*) E. Portera. Zgodnie z tą teorią, zachowania są uzależnione od modelu motywacji. Jednostka dąży do osiągnięcia nagrody zgodnie z charakterystykami jednego z trzech modeli motywacyjnych (tabela 3).

Tabela 3. Modele motywacji wg E. Portera (na podstawie [3])

Opis	Charakterystyka
Motywacja altruistyczno-wspomagająca	
Szukanie nagrody przez wprowadzanie harmonii w kontaktach z innymi i przez przyczynianie się do korzyści dla innych bez zważania na uzyskanie wynagrodzenia za swe zasługi	Ukierunkowanie na ludzi, popieranie, zaufanie, pomoc, wrażliwość na potrzeby innych
Motywacja apodyktyczno-kierująca	
Szukanie nagrody przez pewność siebie i kierowanie działania innych z wyraźnym poczuciem prawa do nagrody	Ukierunkowanie na działanie, ambicja, przekonanie, siła, organizacja
Motywacja analityczno-autonomizująca	
Poszukiwanie nagrody przez osiąganie niezależności i samowystarczalności oraz ładu logicznego	Analiza, dokładność, zasady, niezależność, metodyczność

Znajomość tych charakterystyk pozwala określić typ motywacyjny rozmówcy na podstawie pierwszych kontaktów, a nawet opracowań napisanych przez tę osobę.

Trzecie pytanie, na które trzeba udzielić odpowiedzi pomocnych w planowaniu rozmów, brzmi: jak dużo odbiorca wie na temat, któremu będą poświęcone rozmowy? W celu uzyskania kompletnej i trafnej odpowiedzi należy:

— zasięgnąć opinii innych pracowników;
— przeanalizować wcześniejsze kontakty;
— obserwować pracę rozmówcy;
— określić znaczenie tematyki dla tej osoby;
— zrezygnować z domniemań i wstępnie przedstawić propozycję spotkania wraz z informacją o tematyce rozmowy.

Kiedy zostały rozstrzygnięte kwestie dotyczące postawy rozmówcy, można rozpocząć działania mające na celu pozyskanie jego zainteresowania i zaangażowania. Zalecaną techniką sprzyjającą osiągnięciu tego celu jest *nastawienie-na-ty*, czyli spojrzenie na sytuację z punktu widzenia drugiej osoby[16]. Wyższość zalecanej techniki wynika z następujących jej cech (por.: [Baird'1983]):

[14] Inną dziedziną, która jest przedmiotem wykładów uniwersyteckich w kształceniu profesjonalnym w krajach zachodnich jest etyka zawodowa (ang. *business ethics*).

[15] Według A. Maslowa istnieje hierarchia potrzeb obejmująca 5 klas: *fizjologiczne, bezpieczeństwa, społeczne, uznania i samorealizacji*. Zaspokojenie określonych potrzeb powoduje ujawnienie się potrzeb wyższego rzędu. Niezaspokojenie określonych powoduje, że dominują one działania jednostki. McClelland wymienia potrzeby związane z jakością pracy: potrzeby *osiągnięć, władzy, przynależności* i *uznania*. Williams'1951.

● **Kontakt staje się ważniejszy dla odbiorcy**
Wiadomość przekazywana w zalecanej formie wydaje się pilniejsza i ważniejsza dla osoby, której jest ona przekazywana

nie mów:	powiedz:
World-Soft jest żywotnie zainteresowany poznaniem ocen jakości i przydatności naszego oprogramowania.	**Twoja** opinia jest istotna! Bez **Twoich** pomysłów i opinii nie możemy wiedzieć, czego potrzebujesz.

● **Stwarza się pozytywną atmosferę kontaktów**
Omówiona wcześniej istota pozytywnego nastawienia wyjaśnia znaczenie tej techniki dla dobrej woli odbiorcy. Dobrze znamy bezosobowe, zimne, formalne, a czasami nieprzyjemne formy komunikacji i łatwo jest przewidzieć, jakie odczucie może towarzyszyć kontaktom z akcentem osobistym i pozytywnym nastawieniem.

nie mów:	powiedz:
My w World-Soft oczykujemy, że wszystkie nasze usługi będą świadczone szybko i na najwyższym poziomie.	**Twoja** dbałość o jakość i szybkość świadczonych usług zagwarantuje Ci długą współpracę z naszą firmą.

● **Wskazuje się na korzyści dla odbiorcy**
Przekazy, w których wyraźnie określa się „co to da odbiorcy", w większym stopniu będą interesowały i angażowały odbiorcę. Akcentując w nich jego korzyści zwiększamy szanse:
— otrzymania szybkiej odpowiedzi;
— otrzymania pozytywnej odpowiedzi;
— zainicjowania kontaktów lub działań korzystnych dla nadawcy;
— nawiązania bardziej osobistych kontaktów z odbiorcą.
Wybór wskazywania korzyści odbiorcy musi być oczywiście dokonany z jego punktu widzenia. Wskazanie nieprzemyślanych „korzyści" według uznania nadawcy może mieć skutki odwrotne od zamierzonych. Motywację odbiorcy należy wiązać z:
— korzyściami dotyczącymi wykonywanej pracy (poprawa warunków pracy, ułatwienie rozwiązywania problemów, usunięcie przyczyn zakłóceń, lepsza wydajność, wyższe zarobki itp.);
— korzyściami dla organizacji (zwiększenie zysków, zmniejszenie kosztów, korzystniejsze perspektywy rozwojowe, lepsze kontakty z klientami);
— korzyściami osobistymi (realizacja określonego pomysłu, uznanie ważności jakiegoś celu, promowanie osoby itp.).

Szczególnie skuteczne jest wskazanie wszechstronnych korzyści z zakresu wszystkich wymienionych wyżej grup, przy czym — należy jeszcze raz przypomnieć — muszą one zostać tak określone, by zostały zaakceptowane przez odbiorcę.

Ostatnim związanym z wyborem strategii problemem do rozwiązania jest dobór odpowiedniej formy przekazu (kanału, środka komunikacji). Czynność ta wydaje się zupełnie prosta i pomocny powinien się okazać zdrowy rozsądek. Praktyka wykazuje jednak, iż jest on często lekceważony.

Ze studium przeprowadzonego w licznych organizacjach wynika, że w określonych sytuacjach najkorzystniejsze wyniki przynosi wykorzystanie przekazu ustnego, pisanego, ustnego poprzedzonego pisanym lub pisanego poprzedzonego ustnym [13]. Poniżej zestawiono te sytuacje uzupełniając je o typowe kontakty analityków systemów informacyjnych z użytkownikami (są one oznaczone gwiazdką):

[16] Odmienną postawą jest *nastawienie-na-ja*, przy którym uwzględnia się własny punkt widzenia.

Przekaz pisemny:
1. Komunikaty mające na celu podjęcie przez pracownika działania;
2. Przekazanie informacji o charakterze ogólnym;
★ Przedstawienie celów podejmowanego zamierzenia wszystkim pracownikom lub ich grupie;
★ Prośba o udzielenie informacji adresowana do dużej grupy pracowników.

Przekaz ustny.
1. Upomnienie pracownika za złe wyniki pracy;
2. Wywołanie dyskusji między pracownikami na temat problemów związanych z wykonywaną pracą;
★ Wywiady indywidualne i grupowe;
★ Prezentacje projektów;
★ Doradztwo, konsultacje.

Przekaz ustny — pisemny.
1. Dostarczenie informacji wymagającej natychmiastowego działania ze strony pracownika;
2. Przekazanie polecenia (dyrektywy);
3. Ogłoszenie informacji dotyczącej ważnej zmiany w polityce przedsiębiorstwa;
4. Przedstawienie kierownikowi informacji o postępie prac;
5. Promocja akcji mającej na celu poprawę bezpieczeństwa pracy;
6. Pochwała pracownika za wyróżniające się wykonywanie pracy;
★ Ustalenia dotyczące warunków i organizacji prowadzenia prac projektowych i wdrożeniowych;
★ Informowanie o konieczności zmiany harmonogramu prac i (lub) preliminarza;
★ Przekazanie sugestii dokonania zmian organizacyjnych związanych z wdrożeniem systemu.

Przekaz pisemny — ustny
Ten rodzaj przekazu uznany przez uczestniczących w badaniach za nieskuteczny we wszystkich typowych kontaktach.

Ważne znaczenie dla doboru odpowiedniej formy komunikacji ma przewidywana reakcja na przekaz. Jeśli oczekuje się reakcji neutralnej, celowe jest wykorzystanie notatek, pism okólnych itp. Przekaz, który zgodnie z oczekiwaniami spowoduje negatywną reakcję, wymaga kontaktu osobistego (indywidualnego lub zespołowego). W przypadku oczekiwanej pozytywnej reakcji dobór odpowiedniej formy komunikacji jest mniej znaczący i będzie zależeć od możliwości nadawcy.

Przyjęta forma komunikacji może w dużym stopniu wzmacniać lub uzupełniać treść przekazywanej wiadomości. Bezpośrednia rozmowa dyrektora z pracownikiem, który przygotowuje np. raport zawierający ocenę systemu informacyjnego, będzie służyła przypomnieniu i podkreśleniu wagi przywiązywanej do tego raportu. Przekazanie w formie pisemnej (np. pisma okólnego, notatki) rozstrzygnięć przyjętych podczas zebrania zespołu projektowego, ma na celu podkreślenie ich ważności.

Podstawowym czynnikiem determinującym najkorzystniejszą formę komunikacji jest czas. W warunkach odpowiedniego wyposażenia technicznego i dobrej organizacji czas komunikacji może oznaczać korzyści lub straty i brak efektywności. Spotkania zespołów, narady oraz inne formy przekazu bezpośredniego są bardzo czasochłonne. Przekaz pisemny wymaga krótszego czasu przygotowania, jednak czas odpowiedzi oraz niemożliwość otwartej wymiany myśli i odczuć ogranicza możliwości oddziaływania na odbiorcę. Wybór najkorzystniejszej metody komunikacji powinien być wynikiem oceny strat czasu przy wykorzystaniu danego środka komunikacji w od-

niesieniu do potrzeb wynikających z sytuacji, w jakiej przekaz jest dokonywany.

	Sprzężenie zwrotne	Dwukierunkowość przekazu	Szybkość przesyłania	Możliwość dokumentowania	Zainteresowanie i znaczenie
Listy				+	
Memoranda			+	+	+
Raporty			+	+	+
Podręczniki, przewodniki				+	
Publikacje do użytku wewnętrznego				+	
Biuletyny				+	
Rozmowy telefoniczne	+	+	+		
Rozmowy, wywiady	+	+			
Spotkania zespołów	+	+			
Wystąpienia słowne	+	+			

Rys. 2. Skuteczność form komunikacji (na podstawie /3])

Na rys. 2 przedstawiono ocenę wybranych form komunikacji przy uwzględnieniu omawianych powyżej kryteriów.

Podsumowując rozważania na temat strategii komunikowania przypomnijmy podstawowe pytania, na jakie należy udzielić odpowiedzi przed przekazaniem wiadomości:
— co zamierzam osiągnąć?
— jak mogę zainteresować, zaangażować odbiorcę?
— jak powinienem przekazać komunikat?

Staranne przygotowanie podejmowanego kontaktu w dużym stopniu zwiększa szanse osiągnięcia celu i utrzymania, a nawet poprawienia korzystnego nastawienia odbiorcy.

W drugiej części dokończymy rozważania prezentując podstawowe formy komunikacji.

LITERATURA

[1] Athey T. H., Zmud R. W.: Introduction to Computers and Information Systems. 2nd Ed. Scott, Foresman and Co., Glenview-Boston-London, 1988
[2] Bocchino W. A.: Systemy informacyjne zarządzania. Narzędzia i metody. WNT. Warszawa, 1975
[3] Baird J. W., Stull J. B.: Business Communication. Strategies and Solutions. McGraw-Hill. New York, 1983

Dr Marian **KURAŚ** ukończył w 1970 r. Wyższą Szkołę Ekonomiczną w Krakowie. Przez ponad trzy lata pracował w przemyśle petrochemicznym. W tym czasie ukończył podyplomowe studium projektowania systemów EPD. Od roku 1973 pracuje w Akademii Ekonomicznej w Krakowie, obecnie w Zakładzie Zastosowań Informatyki. W roku 1981 uzyskał doktorat w AE. Jest autorem (współautorem) ponad 80 opracowań naukowych.

dokończenie na s. 15

Uwaga Abonenci Teleksowi!!!

TRADYCYJNY TELEKS JEST JUŻ NIEPOTRZEBNY!!!
PcTelex i Telex-80
TO NAJBARDZIEJ ZAAWANSOWANE
KOMPUTEROWE SYSTEMY TELEKSOWE

★ komfort i szybkość pracy
★ oszczędność ponad 70% czasu w porównaniu z tradycyjnym teleksem
★ treść na ekranie komputera
★ nadawanie i odbiór przez całą dobę niezależnie od komputera
★ własny zasilacz i złącze drukarki
★ dostęp do teleksu z każdego stanowiska w sieci komputerowej
★ obsługa do 4 linii teleksowych
★ najwyższa jakość wykonania
★ polska homologacja

OBA URZĘDZENIA PRACUJĄ JUŻ W WIĘKSZOŚCI KRAJÓW EUROPY ZACHODNIEJ I SZYBKO ZDOBYWAJĄ RYNEK POLSKI

ZAPRASZAMY NA POKAZ DO NASZEJ FIRMY!!!

Producent:
Niels Neesgaard International Aps Dania

Autoryzowany dystrybutor:
BIT — Spółka z o.o., 80-320 Gdańsk-Oliwa, Al. Grunwaldzka 521
telefon 52-45-41, 52-33-41, telefaks 52-03-21, teleks 0512516

SO/549/91

TOMASZ KALINOWSKI
Warszawa

Analizatory wykonania programów

Każdy program ma pewne właściwości statyczne i dynamiczne. O właściwościach statycznych można uzyskać informacje na podstawie kodu źródłowego, raportu translacji lub konsolidacji. Inaczej mówiąc, ich uzyskanie nie wymaga wykonania programu. Zalicza się do nich rozmiar kodu i danych, statycznych graf wywołań, położenie symboli globalnych względem początku programu itd. Do właściwości dynamicznych należą m.in. wykorzystanie procesora, urządzeń zewnętrznych i stronicowanie pamięci. Analizatory wykonania są narzędziami służącymi do uruchamiania, testowania i optymalizacji programów. Wynikiem ich działania jest **raport wykonania** (*ang. execution profile*), przedstawiany w postaci tabeli, histogramu lub grafu. Zastosowanie analizatorów w fazie uruchamiania pozwala wykryć nieskończone pętle i złe warunki rozejścia, a w fazie testowania — znaleźć fragmenty, które w największym stopniu wpływają na czas wykonania programu. Informacje te umożliwiają programiście podjęcie decyzji o zmianie algorytmu lub języka programowania.

Metodologia posługiwania się analizatorem wykonania programów

Analiza wykonania programu może być wykonana na poziomie adresów fizycznych, instrukcji wysokiego poziomu lub podprogramów. Nowoczesne programy składają się z wielu modułów (procedur i funkcji), które zapewniają pewne usługi. Moduły takie mają często różnych autorów, a stosowanie analizatora pomaga ustalić hierarchię wywołań i ocenić wpływ wykonania pewnych operacji na zachowanie całego systemu. Niektóre z analizatorów (np. *gproof* [3]) oprócz wykonania **raportu statycznego** (ang. *static profile, flat profile*) pozwalają przedstawić **graf wywołań** (ang. *call graph*).

```
#include <memory.h>
int insert_sort(void *p_arr, unsigned int arr_range,
                unsigned int item_size, int (*compare)())
{
   int i,j;
   void *ptr;
   if ((ptr = malloc(item_size)) == NULL)
      return(-1);
   for( i = 1; i < arr_range; i++)
   {
      j = i;
      memmove(ptr, (void*) ((char*)p_arr + item_size * i),item_size);
      while ( j > 0 &&
         compare((void*)((char*)p_arr + item_size * (j - 1)), ptr) < 0 )
      {
         memmove((void*)((char*)p_arr + item_size * j),
                 (void*)((char*) p_arr + item_size * (j - 1)),item_size);
         j--;
      }
      memmove((void *)((char*)p_arr + item_size * j), ptr,item_size);
   }
   free(ptr);
   return(0);
```

Rys. 1. Sortowanie przez wstawianie

Mgr inż. Tomasz KALINOWSKI w 1990 r. ukończył studia na Wydziale Elektroniki w Instytucie Informatyki PW. Obecnie studiuje na Wydziale Matematyki, Informatyki i Mechaniki UW. Interesuje się zagadnieniami inżynierii i oprogramowania.

```
#include <stdio.h>
#include <string.h>
#include "ins_sort.h"
typedef struct
{
  char name[8];          /* imie dziecka                    */
  int age;               /* wiek                            */
} child;
                         /* porownanie wieku dzieci         */
int younger(child *left,child *right)
{
   if (left->age > right->age)
     return(-1);
   if (left->age < right->age)
     return(1);
   return(strcmp(right->name, left->name));
}
child children[]={ "ewa"  , 3,
                   "ania" , 2,
                   "ola"  , 7,
                   "magda", 5,
                   "gosia", 5};
main()
{
  int i;
  insert_sort(children,sizeof(children)/sizeof(child),
                       sizeof(child),younger);
  for( i = 0; i < sizeof(children)/sizeof(child);i++)
    printf("%2d  %8s",children[i].age, children[i].name);
}
```

Rys. 2. Program wykorzystujący algorytm sortowania przez wstawianie

Rozpatrzmy przykład funkcji, która sortuje przez wstawianie (rys. 1). Funkcję tę zaimplementowano jako jednostkę rodzajową [6], a przykładowy program realizujący sortowanie wg tego algorytmu przedstawiono na rys. 2. Statyczny raport wykonania dla tego programu mógłby wyglądać jak na rys. 3.

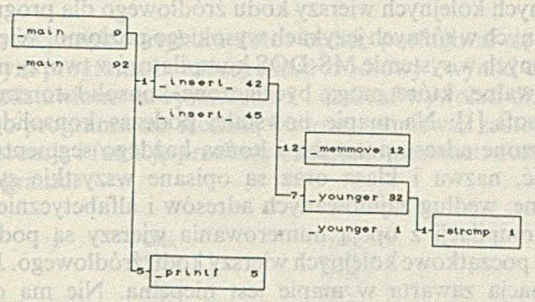
nazwa podprogramu	liczba wykonań	liczba wierszy wykonanych w podprogramie
_main	1	9
_insert_sort	1	42
_memmove	12	12
_younger	7	32
_strcmp	1	1
_printf	5	5

Rys. 3. Raport statyczny dla programu z rys. 2

Taki obraz wykonania jest jednak niekompletny. Raport statyczny nie zawiera informacji na temat przekazywania sterowania w programie. Programista musi posługiwać się kodem źródłowym programu w celu ustalenia hierarchii wywołań.

Graf przedstawiony na rys. 4 obrazuje natomiast zależności między funkcjami. Z lewej strony każdego bloku jest podana liczba wywołań podprogramu. Liczby umieszczone wewnątrz określają **czas własny wykonania** (ang. *self time*), poniżej jest

umieszczony **czas dziedziczony** (ang. *inherited*), będący sumą czasów wykonania podprogramów należących do łańcucha wywołań zapoczątkowanego przez ten podprogram.



Rys. 4. Graf wywołań dla programu z rys. 2

Power [5] proponuje następujący sposób wykorzystania analizatora:
1. Wykonaj program pod nadzorem analizatora.
2. Znajdź na podstawie raportu wykonania najczęściej wykonywane obszary programu.
3. Sprawdź, czy zmiana algorytmu może przyspieszyć wykonanie.
4. Wprowadź poprawki i powtarzaj powyższe czynności tak długo, aż dokonywane zmiany nie będą przynosiły znaczących rezultatów.

Zastosowanie przedstawionej metody postępowania do powyższego przykładu mogłoby wyglądać następująco: jeżeli sortowanie trwałoby dłużej niż się spodziewano, to przyczyną mogą być nieodpowiednie dla tego algorytmu dane (sortowanie przez wstawianie pozwala uzyskać najlepsze wyniki dla ciągów prawie uporządkowanych). Załóżmy, że większość czasu wykonania przypadałaby na funkcję **memmove**. Oznaczałoby to, że należy zastosować algorytm o mniejszej liczbie przestawień, np. sortowanie przez wybór. Jeśli wykonanie funkcji **younger** stanowiłoby przeważającą część sortowania, to należałoby zmniejszyć liczbę porównań np. przez zastosowanie wyszukiwania binarnego w części uporządkowanej.

Zbieranie danych pomiarowych

Wykonanie programu może być reprezentowane jako sekwencja stanów programu, opisywanych za pomocą adresu instrukcji lub nazwy procedury i numeru wiersza. Na podstawie tych informacji można ustalić liczbę wykonanych instrukcji, wywołanych podprogramów lub określić przepływ sterowania. Do podstawowych technik stosowanych przy odczytywaniu stanów programu należą: **śledzenie** (ang. *tracing*) i **próbkowanie** (ang. *sampling*).

W technice śledzenia przejście między kolejnymi stanami jest traktowane jako **zdarzenie pomiarowe**. Zdarzenia pomiarowe są zliczane dla wszystkich stanów programu. Śledzenie nie pozwala mierzyć czasu wykonania, ponieważ okres między kolejnymi odczytami stanu jest bardzo krótki. Stosuje się jedynie przybliżoną miarę czasu, jaką jest liczba wykonanych instrukcji, co powoduje, że na podstawie sporządzonego raportu nie można ustalić np. wpływu obsługi urządzeń wejścia-wyjścia na wykonanie programu. Technika ta jest przydatna zwłaszcza przy uruchamianiu, gdyż użytkownik otrzymuje dokładny raport zawierający liczbę zdarzeń pomiarowych dla każdego stanu. Duża częstotliwość zbierania danych pomiarowych jest jednak nie tylko zaletą, ale również wadą, albowiem znacznie wpływa na zmniejszenie szybkości wykonania.

Próbkowanie polega na cyklicznym przerywaniu wykonania i odczytywaniu stanu programu. Próbkowanie może być **okresowe** lub **probabilistyczne**. Ta druga metoda ma lepsze właściwości statystyczne, gdyż zapobiega przypadkowej synchronizacji wykonywania programu i zbierania próbek. Jest to mało prawdopodobne, ale mogłoby doprowadzić do nieprawidłowych wyników. Gdyby — na przykład — czas wykonania pętli, w której byłaby wywoływana procedura, równał się okresowi próbkowania, to użytkownik mógłby otrzymać informacje, że procedura albo w ogóle nie była wykonywana, albo była wykonywana bardzo często i brak byłoby danych o czasie wykonania pętli.

Ponieważ zbieranie próbek nie jest związane ze zmianami stanu programu, próbkowanie pozwala uwzględniać czas obsługi urządzeń zewnętrznych i stronicowania pamięci. Mniejsza ilość informacji niż w technice śledzenia powoduje, że program może być szybciej wykonywany. Ponieważ otrzymany raport jest niekompletny, więc aby uzyskać wiarygodne wyniki, należy wykonać program wielokrotnie.

Instrumentacja analizowanego programu

Wykonanie analizowanego programu jest zatrzymywane w celu odczytania i zliczania stanów programu, identyfikowanych na podstawie wartości licznika instrukcji. Do odtworzenia stanu na poziomie języka programowania są potrzebne dodatkowe dane o adresach symboli i wierszach kodu. **Instrumentacja** jest to nadanie programowi użytkownika takich właściwości, które umożliwią realizację wybranej techniki pomiarowej i przetworzenie wyników pomiarów do postaci wygodnej dla użytkownika. Może ona być realizowana na poziomie kodu źródłowego, kompilatora, systemu operacyjnego, środowiska uruchomieniowego lub sprzętu.

Na poziomie kodu źródłowego można dokonać instrumentacji przez wstawienie liczników, co jednak powoduje zwiększenie programu nawet o 600% (jak w przypadku stosowania opcji COUNT udostępnianej przez kompilator w PL/I [3]). Wadami tego rozwiązania jest bardzo wolne wykonanie oraz konieczność budowy wielu narzędzi do rozpoznawania składni różnych języków programowania. Stosowanie tej metody wymaga też czasochłonnego przygotowania programu do analizy. Jeżeli ma nastąpić zmiana analizowanego modułu, to należy usunąć z niego liczniki i wstawić je w innym, a następnie skompilować oba moduły i powtórnie skonsolidować system.

Instrumentacja na poziomie kompilatora pozwala uzyskać informacje o adresach początkowych kolejnych wierszy kodu źródłowego oraz wstawić wywołania modułów sterujących w prologach funkcji [3]. Ułatwia to odwzorowanie instrukcji wysokiego poziomu na adresy fizyczne, używane na poziomie procesora. Instrumentacja w czasie wykonania jest pozbawiona wad opracowywania na poziomie kodu źródłowego. System nie wymaga rekompilacji, jest dopuszczalne łączenie różnych języków programowania, a warunki działania są zbliżone do rzeczywistych. Analizator ładuje i inicjuje program wynikowy tworząc środowisko uruchomieniowe, w którym jest możliwe przejęcie sterowania od wykonywanego programu, na przykład przez rozmieszczenie pułapek programowych w punktach rozejścia. Rozwiązania sprzętowe polegają na wykorzystaniu oddzielnego procesora do generowania próbek. Analizator SPY wykorzystuje w tym celu jeden z procesorów peryferyjnych CDC6000 [5]. Monitory sprzętowe zawierają wiele szybkich rejestrów, których zawartość jest zwiększana, jeśli na szynie pojawi się odpowiadający im adres.

Analizator wykonania wprowadza mniejsze opóźnienia czasowe niż program uruchomieniowy. Dane są zbierane w sposób

dynamiczny, co umożliwia łatwiejsze wykorzystanie w systemach czasu rzeczywistego. Analizator może być stosowany w pierwszej fazie uruchamiania do wykrywania wielu prostych błędów, jak nieskończone pętle lub pominięcie istotnych fragmentów programu. Przyczyny wadliwego działania (złe warunki rozejścia w powyższych wypadkach) mogą zostać znalezione przy użyciu programu uruchomieniowego. Zastosowanie profilu dynamicznego w fazie testowania pozwala zoptymalizować program np. przez zmianę wielkości i liczby buforów wejścia-wyjścia lub zmianę algorytmu i języka programowania.

Analizator wykonania programów SEA

Analizator wykonania programów SEA (ang. *simple execution analyzer*) umożliwia sporządzanie raportów na poziomie wierszy kodu źródłowego i budowę grafu wywołań dla programów napisanych w językach C, Pascal i Fortran. Sprzęg użytkownika pozwala wyświetlać struktury programu, zawartość plików i raportów wykonania oraz zapewnia komunikację z systemem operacyjnym (redagowanie wiersza zlecenia, zmianę katalogu i dysku oraz tymczasowe wyjście na poziom interpretera poleceń systemu operacyjnego).

Instrumentacja i zbieranie danych pomiarowych

W analizatorze SEA do zbierania danych pomiarowych wykorzystano mechanizm pułapek. **Pułapki programowe** są to jednobajtowe instrukcje generujące przerwanie. Kod instrukcji, znajdujący się pod adresem, dla którego ma nastąpić zatrzymanie, jest umieszczany w obszarze roboczym programu uruchomieniowego. Użycie jednobajtowych instrukcji ma zapobiec następującej sytuacji [4]:

```
        X:...
          ...
J         RET
J + 1   Y:...
J + 2
```

Jeżeli naszym celem byłoby zatrzymanie wykonania w chwili zakończenia procedury *X*, a wywołanie programu uruchomieniowego lub analizatora byłoby relizowane za pomocą instrukcji skoku, to wywołanie funkcji *Y* spowoduje błąd, gdyż licznik instrukcji zostanie ustawiony na argumencie, a nie na operandzie instrukcji.

Najczęściej pułapki umieszcza się w trakcie wykonywania (co pozwala wybrać miejsce zatrzymania w zależności od stanu programu), ale mogą być one umieszczane przez preprocesor w kodzie źródłowym.

Do zalet pułapek programowych należy zaliczyć prostotę realizacji i brak dodatkowego sprzętu. Niestety mogą być one stosowane jedynie w pamięci RAM. Stosuje się trzy rodzaje pułapek programowych:

- stałe (po zatrzymaniu nie są usuwane);
- dynamiczne (wymagają każdorazowego ustawienia pod danym adresem, jeśli program został zatrzymany przez tę pułapkę);
- z licznikiem (przy przejściu przez pułapkę zmniejsza się wartość licznika; wykonanie programu zostaje zatrzymane, jeśli licznik osiągnie wartość zero).

W analizatorze SEA zbieranie danych przebiega w następujący sposób: po załadowaniu programu są ustawiane pułapki z licznikiem pod adresami początkowymi wierszy kodu źródłowego. Do liczników wpisuje się wartość maksymalną dla typu *integer* lub *long integer*. Po wykonaniu różnica między wartością początkową a końcowym stanem licznika określa liczbę wykonań instrukcji.

Powstaje zatem pytanie, skąd uzyskiwać dane o adresach fizycznych kolejnych wierszy kodu źródłowego dla programów napisanych w różnych językach wysokiego poziomu. Większość dostępnych w systemie MS-DOS kompilatorów tworzy moduły relokowalne, które mogą być łączone konsolidatorem firmy Microsoft [1]. Na mapie powstałej podczas konsolidacji są zaznaczone adresy początku i końca każdego segmentu, jego długość, nazwa i klasa oraz są opisane wszystkie symbole globalne: według zajmowanych adresów i alfabetycznie. Podczas kompilacji z opcją numerowania wierszy są podawane adresy początkowe kolejnych wierszy kodu źródłowego. Jednak informacja zawarta w mapie jest niepełna. Nie ma opisów symboli globalnych, nie jest znana ich klasa, typ i rozmiar. Trudności sprawia określenie bieżąco wykonywanej procedury, gdyż mapa zawiera jedynie adresy początku kodu podprogramów. Dodatkowym ograniczeniem jest brak informacji na temat symboli klasy *static* w języku C oraz procedur i funkcji zadeklarowanych wewnątrz innego podprogramu lub w części implementacyjnej pakietu (Turbo Pascal). W mapie nie są podawane nazwy plików włączanych przez preprocesor, co utrudnia odwzorowanie numerów wierszy i adresów w pamięci.

Zbieranie danych do raportu statycznego może być ograniczone do niektórych plików i podprogramów. Pułapki z licznikiem nie są wówczas ustawiane dla pozostałej części kodu. Analizator umożliwia również wybór między wykonaniem programu a śledzeniem, co łącznie z możliwością ustawiania pułapek stałych pozwala dodatkowo zawęzić zakres sporządzanego raportu. Pułapki stałe mogą znaleźć zastosowanie w sytuacjach:

- wyłączenia śledzenia fragmentów podprogramu (np. pętli o znanej liczbie przebiegów, w której sekwencyjnie są wykonywane instrukcje proste); pułapkę należy wówczas ustawić na początku i końcu tego obszaru; po zatrzymaniu analizy przez pierwszą pułapkę trzeba wyłączyć śledzenie i uruchomić program, a po powtórnym zatrzymaniu kontynuować analizę;
- ograniczenia zbierania danych do wybranych przebiegów pętli; pułapka powinna być ustawiona na początku cyklu; po każdym zatrzymaniu użytkownik musi podjąć decyzję, czy kontynuować wykonanie, czy śledzenie;
- analizy trudnej do przewidzenia sekwencji instrukcji, np. w przypadku śledzenia wykonania między zakończeniem podprogramu a jego powtórnym wywołaniem; pułapki należy wówczas ustawić na początku i na końcu kodu procedury; analizę rozpoczyna się po zatrzymaniu wykonania przez pułapkę umieszczoną pod adresem ostatniej instrukcji procedury, a kończy po powtórnym wywołaniu podprogramu.

```
#include <stdio.h>

rfact_n(int n)
{
  printf("rfact_n %d\n",n);
  if (n > 1)
    return(n*rfact_n(n-1));
  else
    return 1;
}

ifact_n(int n)
{
  int j;
  j =n;
  printf("ifact_n %d\n",n);
  while( n > 1)
    j *= --n;
  return j;
}

fact(int n,int (*fact)())
{
  printf("fact    %d\n",n);
  (*fact)(n);
}

add_n(int n)
{
  printf("add_n   %d\n",n);
  n++;
  if (n < 10)
    mult_n(n);
  if (n%2)
    fact(n,rfact_n);
  else
    fact(n,ifact_n);
}

mult_n(int n)
{
  printf("mult_n  %d\n",n);
  n *=2;
  add_n(n);
}

main()
{
  printf("main\n");
  add_n(1);
  rfact_n(3);
  ifact_n(3);
}
```

Rys. 5. Przykładowy program

Wykonanie programu, dla którego ma być zbudowany dynamiczny graf wywołań, poprzedza umieszczenie pułapek stałych na początku kodu procedur i funkcji oraz pułapek z licznikiem dla wszystkich wierszy kodu.

W dalszej części artykułu autor będzie posługiwał się przykładowym programem, którego kod źródłowy i wyniki działania przedstawiono na rysunkach 5 i 6. Graf wywołań dla tego prostego programu ma złożoną budowę, ponieważ zastosowano rekurencję i funkcję przekazywaną jako parametr formalny podprogramu.

```
main          rfact_n 3
add_n   1     rfact_n 2
mult_n  2     rfact_n 1
add_n   4     fact    5
mult_n  5     rfact_n 5
add_n   10    rfact_n 4
fact    11    rfact_n 3
rfact_n 11    rfact_n 2
rfact_n 10    rfact_n 1
rfact_n 9     fact    2
rfact_n 8     ifact_n 2
rfact_n 7     rfact_n 3
rfact_n 6     rfact_n 2
rfact_n 5     rfact_n 1
rfact_n 4     ifact_n 3
```

Rys. 6. Wyniki działania programu z rys. 5

Po zatrzymaniu przez pułapkę stałą dla procedur i funkcji oblicza się liczbę instrukcji wykonanych od ostatniego punktu przerwania. Liczba ta jest dodawana do liczby instrukcji dziedziczonych przez podprogramy należące do łańcucha wywołań, do którego zostaje następnie dołączony podprogram wywołujący. Pod adresem powrotu wywołanego podprogramu jest ustawiana pułapka tymczasowa.

Zatrzymanie wykonania przez pułapkę tymczasową powoduje usunięcie procedury z łańcucha wywołań, dodanie liczby instrukcji wykonanych od ostatniej pułapki do liczby instrukcji dziedziczonych przez podprogramy należące do tego łańcucha oraz ustawienie pułapki tymczasowej pod adresem powrotu bieżącej procedury.

Powyższe rozwiązanie nie uwzględnia nielokalnych skoków w uruchamianym programie. W języku C obsługa wyjątków może być realizowana za pomocą funkcji *setjmp* i *longjmp* [6]. Wywołanie funkcji *setjmp* powoduje zapamiętanie stanu stosu. Po wywołaniu funkcji *longjmp* stan ten zostaje odtworzony, a sterowanie jest przekazywane do instrukcji następującej po wywołaniu *setjmp*. Użycie funkcji *longjmp* powoduje, że z łańcucha wywołań nie są usuwane funkcje, których wykonanie zostało rozpoczęte między wywołaniem funkcji *setjmp* a *longjmp* (funkcje są usuwane z łańcucha podczas obsługi pułapki tymczasowej, po zakończeniu wykonania podprogramu).

Utworzenie dynamicznego grafu wywołań wymaga przechowywania liczby wywołań podprogramów oraz liczby aktywacji

```
          ProcCounters  ProcInherited  ActiveCalls
rfact_n         0             0            0
ifact_n         0             0            0
fact            0             0            0
add_n           5             4            1
mult_n          4             0            1
main            2             0            1

CallMatrix
          rfact_n ifact_n fact  add_n  mult_n main
rfact_n     0       0      0      0      0      0
ifact_n     0       0      0      0      0      0
fact        0       0      0      0      0      0
add_n       0       0      0      0      1      0
mult_n      0       0      0      1      0      0
main        0       0      0      1      0      0
```

Rys. 7. Stan struktur danych po powtórnym wywołaniu funkcji add_n dla programu z rys. 5

poszczególnych krawędzi grafu. Graf zaimplementowano w postaci macierzy o rozmiarze $x \times n$, przy czym n oznacza liczbę podprogramów. Znaczna część tej macierzy pozostaje niewykorzystana, ponieważ w programach praktycznie nie występują odwołania między wszystkimi procedurami i funkcjami. Jednak używanie dynamicznych struktur danych w programach uruchomieniowych nie jest wskazane, ponieważ niektóre programy mogą wymagać ciągłych bloków pamięci o znacznym rozmiarze.

Na rysunku 7 przedstawiono stan struktur danych przykładowego programu po drugim wywołaniu funkcji *add_n*.

Tablica *ProcCounters* zawiera liczbę instrukcji wykonanych dla każdego podprogramu. W tablicy *ProcInherited* są przechowywane liczby instrukcji dziedziczonych, a tablica *ActiveCalls* obrazuje łańcuch wywołań. Dla funkcji *main* liczba dziedziczonych instrukcji jest sumą instrukcji wykonanych w funkcji

```
          rfact_n ifact_n fact  add_n  mult_n main
rfact_n     16      0      0      0      0      0
ifact_n      0      0      0      0      0      0
fact         2      1      0      0      0      0
add_n        0      0      3      0      2      0
mult_n       0      0      0      2      0      0
main         1      1      0      1      0      0
```

Rys. 8. Graf wywołań dla programu z rys. 5 po zakończeniu wykonania programu

```
FILES                                  SUBROUTINES
Name        Line Occ.      %    Name      Calls   Line Occ.      %
SIL_C.C          164 100.00
                              _rfact_n         19          95 57.93
                              _ifact_n          2          18 10.98
                              _fact             3          12  7.32
                              _add_n            3          23 14.02
                              _mult_n           2          10  6.10
                              _main             1           6  3.66
```

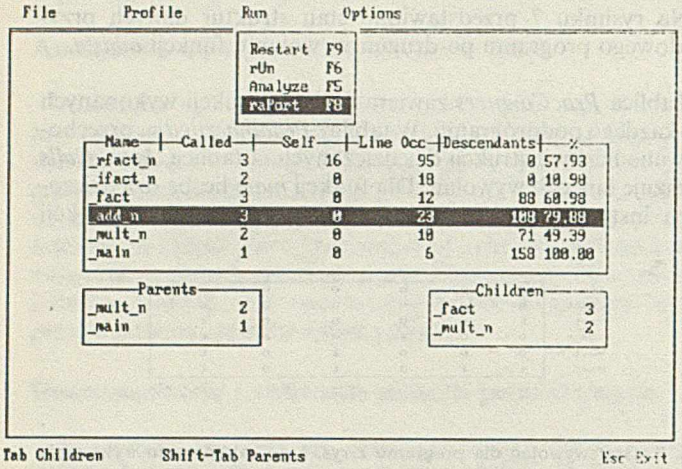
```
SIL_C.C
    #include <stdio.h>

19  rfact_n(int n)
    {
19    printf("rfact_n %d\n",n);
19    if (n > 1)
16      return(n*rfact_n(n-1));
      else
 3      return 1;
19  }

 2  ifact_n(int n)
    {
      int j;
 2    j =n;
 2    printf("ifact_n %d\n",n);
 5    while( n > 1)
 3      j *= --n;
 2    return j;
 2  }
    /* */
 3  fact(int n,int (*fact)())
    {
 3    printf("fact    %d\n",n);
 3    (*fact)(n);
 3  }

 3  add_n(int n)
    {
 3    printf("add_n   %d\n",n);
 3    n++;
 3    if (n < 10)
 2      mult_n(n);
 3    if (n%2)
 2      fact(n,rfact_n);
      else
 1      fact(n,ifact_n);
 3  }

 2  mult_n(int n)
    {
 2    printf("mult_n  %d\n",n);
 2    n *=2;
 2    add_n(n);
 2  }

 1  main()
    {
 1    printf("main\n");
 1    add_n(1);
 1    rfact_n(3);
 1    ifact_n(3);
 1  }
```

Rys. 9. Raport wykonania programu z rys. 5

add_n i *mult_n*. Dla funkcji *add_n* dziedziczona jest jedynie liczba instrukcji wykonanych w funkcji *mult_n*. Graf wywołań jest reprezentowany w tablicy *CallMatrix*. Na rysunku 8 przedstawiono zawartość tablicy *CallMatrix* po zakończeniu wykonania analizowanego programu.

Prezentacja wyników

Po zakończeniu wykonania raport może być wyświetlany na ekranie lub drukowany do pliku. Raport dla wiersza kodu



Rys. 10. Raport wykonania programu z rys. 5 przedstawiony na ekranie

FILES			SUBROUTINES			
Name	Line Occ.	%	Name	Calls	Line Occ.	%
SORT_P.PAS	90	100.00				
			YOUNGER	7	33	36.67
			INSERT_S	1	48	53.33
			0	1	9	10.00

SORT_P.PAS

```
        program sort;
        const all_kids = 5;
        type child = record
                       name:string[8];
                       age:integer;
                     end;
             children = array[1..all_kids] of child;
        const kindergarten :children  = ((name:'ewa' ; age:3),
                                         (name:'ania' ; age:2),
                                         (name:'ola'  ; age:7),
                                         (name:'magda'; age:5),
                                         (name:'gosia'; age:5));


        function younger(left,right:child):boolean;
    7   begin
    7     if left.age > right.age then
    3       younger:= false
          else
    4       if left.age < right.age then
    3         younger:= true
            else
    1         if left.name > right.name then
    1           younger := false
              else
    0           if left.name >= right.name then
    0             younger := true
    7   end;
        procedure insert_sort(var names:children);
        var i,j:integer;
            kid:child;
    1   begin
    1     i:=2;
    5     while  i <= all_kids do
          begin
    4       j := i;
    4       kid := names[i];
    8       while ( j > 1) and not younger(names[j-1],kid) do
            begin
    4         names[j] := names[j-1];
    4         j:= j-1;
    4       end;
    4       names[j] := kid;
    4       i := i+1;
    4     end;
    1   end;
        var i:integer;


    1   begin
    1     insert_sort(kindergarten);
    1     for i := 1 to all_kids do
    5       writeln(kindergarten[i].age:2,'  ',  kindergarten[i].name);
    1   end.
```

Rys. 11. Raport wykonania dla programu wykorzystującego algorytm sortowania przez wstawianie, zakodowanego w języku Pascal

źródłowego oraz procedur i plików przedstawiono na rysunku 9. Dla podprogramów jest podawana liczba wywołań i wykonanych instrukcji oraz liczba określająca, jaki procent wszystkich instrukcji wykonanych przez program stanowi procedura lub funkcja. Zbiór informacji o plikach jest podobny (brak jedynie liczby wywołań).

Najlepszym rozwiązaniem byłoby wyświetlenie grafu wywołań, ale jest to niemożliwe ze względu na dwuwymiarową naturę urządzeń zewnętrznych. Nie można zakładać, że graf jest planarny, nawet gdyby było tak w konkretnym przypadku [3]. Dlatego też graf jest wyświetlany w postaci trzech okien (rys. 10). W środkowym oknie jest umieszczona tablica podprogramów zawierająca:

- nazwę podprogramu,
- liczbę wywołań,
- liczbę wywołań rekurencyjnych,
- liczbę wykonanych instrukcji,
- liczbę instrukcji wykonanych w procedurach potomnych,
- procent, jaki stanowi liczba instrukcji wykonanych w podprogramie w stosunku do całego programu.

W lewym i prawym oknie znajdują się nazwy przodków (potomków) i liczby aktywacji poszczególnych krawędzi.

FILES			SUBROUTINES			
Name	Line Occ.	%	Name	Calls	Line Occ.	%
SORT_F.FOR	110	100.00				

SORT_F.FOR

```
            program sort
            character*8 name(5)
            integer age(5)
      1     name(1)='ewa'
      1     age(1)=3
      1     name(2)='ania'
      1     age(2)=2
      1     name(3)='ola'
      1     age(3)=7
      1     name(4)='magda'
      1     age(4)=5
      1     name(5)='gosia'
      1     age(5)=5
      1     call isort(name,age)
      1     do 10 i = 1,5
      5  10 write(*,20) age(i),name(i)
         20 format(1x,i2,1x,a8)
      1     stop
      0     end

            logical function young(name1,age1,name2,age2)
            character*8 name1,name2
            integer age1,age2
      7     if (age1.gt.age2) then
      3     young = .false.
      3     else
      4     if (age1.lt.age2) then
      3     young = .true.
      3     else
      1     if (name1.gt.name2) then
      1     young = .false.
      1     else
      0     if (name1.le.name2) then
      0     young = .true.
            endif
            endif
            endif
            endif
      7     return
      0     end

            subroutine isort(name,age)
            character*8 name(5),kidn
            integer age(5),kida
            logical young
      1     i=2
      5   1 if(i.le.5) then
      4     j = i
      4     kidn = name(i)
      4     kida = age(i)
      8   2 if ((j.gt.1).and. .not. young(name(j-1),age(j-1),kidn,kida)) then
      4     name(j) = name(j-1)
      4     age(j) = age(j-1)
      4     j = j-1
      4     goto 2
            endif
      4     name(j) = kidn
      4     age(j) = kida
      4     i = i+1
      4     goto 1
            endif
      1     return
      0     end
```

Rys. 12. Raport wykonania dla programu wykorzystującego algorytm sortowania przez wstawianie, zakodowanego w języku Fortran

Działanie analizatora sprawdzano nie tylko dla programów napisanych w języku C, ale również w Pascalu i Fortranie. Na rysunkach 11 i 12 przedstawiono raporty wykonania dla programów wykorzystujących algorytm sortowania przez wstawianie, przedstawiony na początku artykułu (rysunki 1 i 2). Dla Fortranu jest możliwe jedynie sporządzania raportów statycznych na poziomie wierszy kodu źródłowego, ponieważ dane o podprogramach nie są odnajdywane w mapie przez moduł obsługi informacji symbolicznej.

* * *

W artykule wykorzystano niektóre wyniki badań przeprowadzonych w ramach pracy magisterskiej *Projekt, realizacja i zastosowanie pakietu wspomagającego budowę programów uruchomieniowych*, wyróżnionej w konkursie PTI za najlepszą pracę magisterską z dziedziny informatyki.

W tym miejscu chciałbym podziękować mgr. inż. R. Rybusowi za inspirację w wyborze tematu pracy i cenne wskazówki przy przygotowaniu koncepcji badań oraz promotorowi mgr.inż. T.Byzi za pomoc w opracowaniu ich wyników.

LITERATURA

[1] Code View and Utilities for the MS-DOS Operating System. Microsoft Corporation, 1987
[2] Gondzio M.: Microprocessor Debugging Techniques and Their Application in Debugger Design. Software Practice & Experience. March 1987, pp. 215 - 226
[3] Graham S. L., Kessler P. P., McKusick M. K.: An Execution Profiler for Modular Programs. Software Practice & Experience, 1983, pp. 671 - 685
[4] Hughes T. P., Sawin D. H.: Breakpoint Design for Debugging Microprocessor Software. Computer Design, November 1978, pp. 99 - 107
[5] Power L. R.: Design and use of a program execution analyzer. IBM Systems Journal, No. 3, 1983, pp. 271 - 291
[6] Rybus R.: Nowoczesne programowanie w języku C. Informatyka Nr 9, 1988, pp. 15 - 18.

Problemy komunikacji...

dokończenie ze s. 9

[4] Daniels A., Yeates D.: Podstawy analizy systemów. WNT. Warszawa, 1974
[5] DeMarco T.: Structured Analysis and System Specification. Prentice-Hall. Englewood Cliffs (NJ), 1979
[6] Dowling W. F., Sayles L. R.: How Managers motivate People: The Imperatives of Supervision. McGraw-Hill. New York, 1978
[7] Gane Ch., Sarson T.: Structured System Analysis. Prentice-Hall, Englewood Cliffs (NJ), 1979
[8]Gasparski W. (red.): Projektoznawstwo. WNT, Warszawa, 1988
[9] Gibb J. R.: Defensive Communication. Journal of Communication, Vol. 11, 1961, pp. 141 - 148
[10] Graef M., Greiller R.: Organization and Management of a Computer Center. Macmillan Publishing Company. New York — London, 1985
[11] Graham J.: Analiza systemowa w jednostkach gospodarczych. PWE, Warszawa, 1975
[12] Kisielnicki J.: Metody informatyczne. PWE, Warszawa 1981
[13] Level D.: Communication Effectiveness. Methode and Situations. ABCA Journal, Fall 1972
[14] Lundeberg M., Goldkuhl G., Nilsson A.: Information Systems Development. A Systematic Approach. Prentice-Hall, Englewood Cliffs (NJ), 1981
[15] Miller D.: O zagadnieniach komunikowania się w trakcie działalności zawodowej (na przykładzie działalności projektantów). Prakseologia, nr 3 - 4, 1984
[16] Miller D.: Wpływ wiedzy projektanta na formułowanie problemu projektowego. Ujęcie prakseologiczne. Ossolineum, Wrocław-Warszawa-Kraków-Gdańsk-Łódź, 1990
[17] Orr K.: Structures Requirements Definition. Topeka, 1981
[18] Podręcznik analizy i projektowania systemów ICL. HiL, Kraków, 1973
[19] Powers M. J., Adams D. R., Mills H. D.: Computer Information System Development: Analysis and Design. South-Western Publishing Co., Cincinatti, Ohio, 1984
[20] Ross D. T.: Structured Analysis (SA): A Language for Communicating Ideas. IEEE Trans. of Software Engineering, Vol. SE-3, No. 1, 1977
[21] Schoppenhauer A.: Erystyka, czyli sztuka prowadzenia sporów. Wydawnictwo Literackie. Kraków, 1976
[22] Walentynowicz B.: Komunikowanie się ludzi w nauce i technice. Prasa Techniczna, nr 1, 1984
[23] Yourdon E.: Modern Structured Analysis. Prentice-Hall, Englewood Clifs (NJ), 1989.

TERA Spółka Z O. O.

Przedsiębiorstwo Popierania Postępu TERA Spółka z o.o. uprzejmie informuje, że posiadając kilkuletnie doświadczenie w instalacji systemów wspomagających zarządzanie

o f e r u j e :

- opracowanie projektów systemów,
- optymalny kosztowo i rozwojowo dobór sprzętu i oprogramowania,
- instalację systemu u klienta,
- bezpłatnie przez rok konserwację oprogramowania oraz serwis sprzętu wraz z doradztwem techniczno-eksploatacyjnym,
- szybką dostawę uzupełnień konfiguracji lub sprzętu mikrokomputerowego niezależnego od systemu, w tym mikrokomputerów ALR firmy Wearnes Technology.

Wszelkie dodatkowe informacje uzyskają Państwo codziennie oprócz niedziel w Biurze Handlowym
40–025 Katowice, ul. 27 Stycznia 31/17
tel. (faks): 51-61-65, teleks: 315448 tera pl

PAMIĘTAJ! Instalacje XENIX/NOVELL/PC MOS 386 oraz serwis to nasza specjalność – ZAPRASZAMY, ponieważ czterech lat doświadczeń nigdzie nie kupisz.

0/19/90

81-745 SOPOT, ul. Polna 1
tel. 51-17-78, 51-64-38, fax 51-64-37, tlx 0512477

computer studio kajkowski ™

OFICJALNY PRZEDSTAWICIEL FIRM:

ORACLE® relacyjna baza danych (wykorzystuje wieloprocesorowość)

digital wieloprocesorowe mikrokomputery i minikomputery RISC

SO/512/91

POZNAJ SIŁĘ NASZYCH SYSTEMÓW INFORMATYCZNYCH!

JUNISOFTEX Sp. z o.o.
44-100 Gliwice ul. Konstytucji 11
Tel. 31-75-10, 31-90-81 do 88 wewn. 250, 272, telefaks: 31-75-10, teleks 036233

Już od 3 lat oferuje kompleksową komputeryzację przedsiębiorstw, a zwłaszcza:

- profesjonalne nowoczesne systemy informatyczne oparte na kompilatorze języka Turbo Pascal 5.5 firmy BORLAND (typu multiusers), pracujące w sieciach NOVELL 2.15 i 3.1 i wykorzystujące interfejs BTRIEVE amerykańskiej firmy SoftCraft – najefektywniejszy i najszybszy system obsługi dużych i bardzo dużych baz danych (do 4 miliardów rekordów w pojedynczej bazie danych).

AUTORYZOWANY RESELLER FIRMY NOVELL!

Polecamy szczególnie systemy zintegrowane

- wielodostępny system finansowo-księgowy wg trójstopniowej analityki w podstawowym układzie: xxx-xxxx-xxxx-xxxxxx z możliwością graficznej prezentacji wyników,
- wielodostępny system gospodarki materiałowej wykorzystujący w strukturze indeksowej SWW i KTM z automatycznym wyliczaniem cyfry kontrolnej (długość indeksu materiałowego 13 + 3 cyfry),
- wielodostępny system gospodarki wyrobami gotowymi wykorzystujący w strukturze indeksowej SWW i KTM,
- wielodostępny system gospodarki przedmiotami nietrwałymi wykorzystujący w strukturze indeksowej SWW i KTM,
- wielodostępny system środków trwałych wykorzystujący klasyfikację środków trwałych GUS,
- wielodostępny system fakturowania sprzedaży,
- wielodostępny system kadrowo-płacowy.

Wszystkie oferowane systemy są zintegrowane ze sobą zgodnie z zasadami rachunkowości księgowej przedstawionymi w miesięczniku RACHUNKOWOŚĆ nr 9, 1990.

Oferujemy również:

- sprzęt komputerowy (sprawdzony w eksploatacji) firm:
  - ALR (Advanced Logic Research) i Storage Dimensions z USA,
  - WEARNES TECHNOLOGY, i inne z Singapuru,
- dostawę i instalację licencjonowanej sieci Novell 2.2 i 3.11,
- usługi w zakresie opracowywania zakładowych indeksów materiałowych wg klasyfikacji GUS (SWW i KTM).

Dysponujemy:

- Szkołą Informatyczną w ramach której organizujemy kursy doszkalające dla kadry kierowniczej przedsiębiorstw, szkolenia w zakresie obsługi naszych systemów aplikacyjnych,
- laboratorium systemu sieciowego NOVELL.

Organizujemy:

- przy współpracy ze Szkołą Komputerową „IMPULS" kursy: DOS, Word Perfect, Pascal, Podstawy obsługi systemu operacyjnego NetWare firmy NOVELL; szkolenia prowadzone są przez doświadczonych wykładowców Politechniki Śląskiej Wydziału Informatyki i Automatyki;
- prezentację naszych systemów w ww. szkołach a także u naszych kontrahentów.

Gwarantujemy:

- najwyższą jakość naszych systemów i usług,
- najszybsze w Polsce wdrożenia systemów (od 1 do 3 miesięcy).

W przypadku niedotrzymania przez nas terminów zwracamy 25% poniesionych nakładów za każdy miesiąc zwłoki. Dla klientów zamawiających kompleksowe usługi – znaczne zniżki cenowe.

Poszukujemy niezawodnych

- pośredników i dealer'ów naszych systemów informatycznych.

Z siły naszych systemów skorzystały dotąd następujące firmy:

Huta ŁAZISKA w Łaziskach Górnych	tel. 24-14-06
Walcownie Metali w Czechowicach-Dziedzicach	tel. 52-35-1
Centrala Zaopatrzenia Górnictwa w Bytomiu	tel. 81-92-91
Mikołowska Fabryka Transformatorów w Mikołowie	tel. 26-25-41
BPiKD PREFAMET-ZREMB w Gliwicach	tel. 38-20-51
BPiKD METALCHEM w Gliwicach	tel. 31-56-33
Instytut Informatyki Teoretycznej i Stosowanej PAN w Gliwicach	tel. 31-71-50
Instytut Spawalnictwa w Gliwicach	tel 31-00-11
PP ENERGOROZRUCH w Gliwicach	tel. 37-66-25
Fabryka Palenisk Mechanicznych w Mikołowie	tel. 26-20-02
Zakłady Mechaniczne Górnictwa i Energetyki WIROMET w Mikołowie	tel. 26-23-46
Przędzalnia Bawełny PRZYJAŹŃ w Zawierciu	tel. 22-39
PRT TERMOIZOLACJA w Zabrzu	tel. 71-40-21
Zakład Informatyki i Techniki Komputerowej MEGA w Gliwicach	tel. 31-52-42
Mikołowska Fabryka Maszyn Górniczych MIFAMA w Mikołowie	tel. 26-22-66
Zakłady Produkcji Urządzeń Mechanicznych ELWO w Pszczynie	tel. 30-61
Zakłady Sprzętu Sieciowego i Elektroinstalacyjnego POLAM w Katowicach	tel. 52-80-31
Zakłady Elektromaszynowe CELMA w Cieszynie	tel. 21-58-1
PSS SPOŁEM w Zawierciu	tel. 21-46-17

Chcesz mieć pełną gwarancję i niezawodne efekty komputeryzacji swojego przedsiębiorstwa – współpracuj z firmą JUNISOFTEX!

0/1/91

TOMASZ KOKOWSKI
MAREK REFORMAT
MACIEJ STROIŃSKI
Instytut Informatyki
Politechnika Poznańska

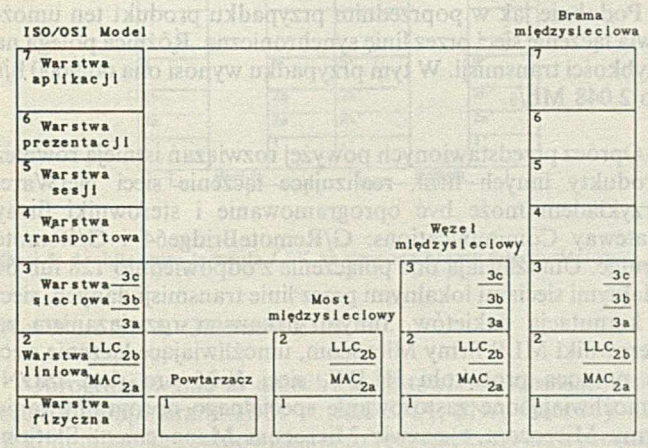
Łączenie lokalnych sieci komputerowych

Obserwowany w ostatnich latach rozwój sieci komputerowych zrewolucjonizował wymianę informacji w świecie. Większość państw dysponuje rozbudowanymi rozległymi sieciami komputerowymi, które połączone ze sobą umożliwiają dostęp do przechowywanej w sieciach informacji i zapewniają usługi (np. pocztę elektroniczną), obejmujące swoim zasięgiem prawie cały świat. Obok nich w dużym tempie rośnie liczba instalacji lokalnych sieci komputerowych obejmujących jedną instytucję lub jej część. Użytkownicy tych sieci coraz częściej wykazują potrzebę korzystania z usług sieci rozległych oraz łączenia się z innymi użytkownikami sieci lokalnych. W tej sytuacji naturalne jest dążenie do łączenia sieci lokalnych z sieciami rozległymi w jeden system będący „siecią sieci". Równocześnie obserwuje się tendencję do rozszerzania zakresu topologicznego sieci lokalnych oraz łączenia różnych typów sieci w zintegrowaną sieć lokalną danej instytucji.

W związku z tym, oceniając lokalne sieci komputerowe należy brać pod uwagę dostępność w tej sieci produktów umożliwiających rozszerzanie zakresu sieci i jej łączenie z sieciami rozległymi. W artykule omówiono tę problematykę w odniesieniu do najbardziej popularnych sieci lokalnych.

Zasady łączenia sieci lokalnych

Celem łączenia lokalnych sieci komputerowych LSK jest realizowane przede wszystkim zwiększenie obszaru obejmowanego przez daną sieć lokalną, budowanie sieci o różnych mediach transmisyjnych oraz połączenie odległych LSK. Ponadto zastosowanie elementów łączących sieci pozwala zdefiniować segmenty odpowiadające logicznym podziałom sieci lub dokonać separacji między częściami sieci.



Rys. 1. Klasyfikacja urządzeń łączących LSK

Na rysunku 1 przedstawiono klasyfikację funkcjonalną urządzeń LSK. Wyróżnia się następujące urządzenia:

- powtarzacz (ang. *repeater*),
- most międzysieciowy (ang. *bridge*),
- węzeł międzysieciowy (ang. *router*),
- brama międzysieciowa (ang. *gateway*).

W klasyfikacji tej istnieje niejednoznaczność w definiowaniu zakresu funkcjonalnego urządzeń powtarzacz i most międzysieciowy [4, 6].

Powtarzacz łączy segmenty sieci na poziomie fizycznym (warstwa pierwsza modelu ISO/OSI). W sieciach połączonych powtarzaczem wszystkie stanowiska sieciowe odbierają wszystkie wysyłane ramki. Urządzenie to zwiększa zasięg sieci przez wzmocnienie sygnału. Najnowsze wersje powtarzaczy zawierają rozbudowaną logikę, która umożliwia budowanie drzewiastej topologii sieci z różnymi rodzajami medium (ośrodków transmisji). Taki rozszerzony zakres funkcjonalny powtarzaczy zaciera różnice między nimi a mostami.

Most międzysieciowy łączy sieci na poziomie łącza danych (protokołu dostępu do medium), tj. w drugiej warstwie modelu odniesienia ISO-OSI. Jest to urządzenie, które częściowo separuje oba segmenty przenosząc ramki z jednego segmentu w drugi tylko wówczas, gdy są one przeznaczone do stanowiska o adresie należącym do drugiego segmentu. Urządzenie to nie ma wydzielonego adresu w sieci. Jest ono transparentne. Mosty mogą łączyć sieci o różnych mediach i protokołach dostępu [1]. Przyjmuje się, że powyżej protokoły są identyczne, np. ISO 8802/2. Mosty mogą być lokalne lub zdalne, co oznacza lokalne lub zdalne połączenie z jedną lub więcej LSK na tych samych zasadach logicznych.

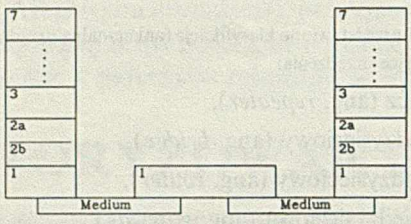
Węzeł międzysieciowy realizuje połączenie sieci w warstwie trzeciej — sieciowej. W każdym stanowisku musi być zaimplementowany protokół międzysieciowy, np. DOD IP [5] lub ISO IP [3]. Węzeł ma adres międzysieciowy, jest więc możliwe przesyłanie informacji związanej z protokołem sieciowym dotyczącej konfiguracji sieci oraz określenie drogi przesyłania danych. W przypadku łączenia lokalnych sieci komputerowych z sieciami rozległymi bardzo często są używane urządzenia łączące w sobie funkcje mostów oraz węzłów międzysieciowych, tzw. „węzeł-most" (ang. *brouter*).

Brama międzysieciowa umożliwia łączenie sieci heterogenicznych (o różnych zestawach protokołów) na poziomie warstwy aplikacji modelu ISO-OSI. Aby umożliwić przesyłanie danych między sieciami, dekoduje się dane zgodnie z protokołami jednej z sieci, a następnie koduje je zgodnie z protokołami drugiej z nich.

Łączenie lokalnych sieci komputerowych

Zastosowanie powtarzaczy

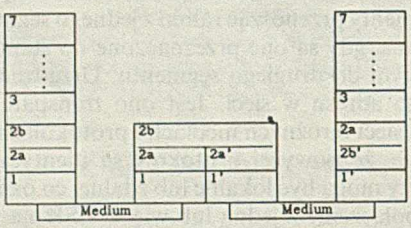
Głównym celem zastosowania powtarzaczy jest rozszerzenie zakresu topologii sieci powyżej odległości jednego segmentu. Architekturę takiego połączenia przedstawiono na rys. 2. Nowoczesne konstrukcje powtarzaczy, np. MultiConnect firmy 3Com, umożliwiają budowanie sieci Ethernet z użyciem różnych ośrodków transmisji, takich jak: „cienki" Ethernet (10BASE2), „gruby" Ethernet (10BASE5), skrętka (10BASE-T) i światłowód (10BASE-F) w strukturze drzewiastej. Odległość między skrajnymi stanowiskami sieci może wówczas wynosi do 4 km.



Rys. 2. Architektura łączenia LSK za pomocą powtarzacza

Zastosowanie mostów międzysieciowych

Architekturę łączenia sieci za pomocą mostów przedstawiono na rys. 3. Właściwość mostu polegająca na separacji łączonych segmentów ma wpływ na przepustowość sieci. Podział sieci na segmenty połączone mostem powoduje podział ruchu w sieci, co oznacza zwiększenie przepustowości w jej segmentach. Podział ten ma również wpływ na ograniczenie efektu uszkodzeń. Uszkodzenie w jednym z segmentów nie wpływa na pracę innych, dopóki nie nastąpi próba przesłania ramki do uszkodzonego segmentu.



Rys. 3. Architektura łączenia LSK za pomocą mostu międzysieciowego

Możliwości funkcjonalne realizacji mostów różnią się między sobą. Niektóre z nich mają funkcje zacierające różnice między mostami i węzłami międzysieciowymi. Mówiąc o realizacji funkcji mostów należy wspomnieć o mostach wewnętrznych i zewnętrznych. Są one związane z systemem Net Ware i dotyczą miejsca wykonywania funkcji mostu. Most wewnętrzny jest związany ze stanowiskiem usługowym. Powstaje przez skonfigurowanie tego stanowiska z więcej niż jednym sterownikiem LSK (maksymalnie cztery). Jest on w pełni transparentny dla użytkowników tych sieci. Most zewnętrzny ma takie same możliwości i jest uruchamiany w stanowisku roboczym w trybie dedykowanym lub niededykowanym.

Istnieją rozwiązania, np. firmy 3Com, w których mosty są wolno stojącymi urządzeniami, przeznaczonymi do łączenia określonych rodzajów sieci, np. IB/2000 umożliwia łączenie sieci 10BASE2 z siecią 10BASE5, a most IB/1 sieci 10BASE5 z siecią 10BROAD36 (ang. *broadband*).

Dotychczas omówione mosty nazywane są lokalnymi. Za pomocą mostów jest również możliwe łączenie sieci znacznie oddalonych od siebie. Mosty te są nazywane mostami zdalnymi. Połączenia mogą być realizowne z zastosowaniem linii telefonicznych i modemów, sieci z komutacją pakietów, linii bezpośrednich lub przez połączenie T1 (kanał PCM 2.048 Mb)s). Poniżej przedstawiono przykładowe produkty służące do łączenia sieci NetWare.

NetWare Link/Asynch

Umożliwia połączenie kilku sieci lokalnych. Użytkownicy mają dostęp do danych i usług przyłączonych sieci. Oprogramowanie mostu jest standardowym wyposażeniem systemu Advanced NetWare v.2.15 oraz SFT NetWare v.2.15. Połączenie sieci jest realizowane przez modem asynchroniczny współpracujący ze sprzęgiem COM1/COM2 w moście zewnętrznym, z kartą WNIM+ (*Wide Area Network Interface Module* — komunikacyjny koprocesor z czterema portami asynchronicznymi) zainstalowaną w stanowisku usługowym lub moście zewnętrznym.

NetWare Link/X.25

Oprogramowanie to umożliwia łączenie sieci NetWare przez sieć z komutacją pakietów X.25. Pojedyncze połączenie z siecią X.25 zapewnia możliwość korzystania z zasobów innych, przyłączonych do sieci X.25 sieci lokalnych. Przyłączenie do sieci z komutacją pakietów jest realizowane przez modem synchroniczny połączony z mostem międzysieciowym za pomocą karty Novell X.25 Adapter (lub Novell X.25 Extended Adapter).

NetWare Link/64

W rozwiązaniu tym łączenie zdalnych sieci lokalnych odbywa się przez linię synchroniczną (z szybkością od 9600 b/s do 64 Kb/s). Oprogramowanie to można zainstalować w stanowisku usługowym lub roboczym. Umożliwia transmisję danych protokołów SPX i novellowskiego NetBIOS-u. Oprogramowanie Link/64 może współpracować z jednym z następujących sprzęgów: V 35, RS 422 lub RS 232. W moście musi być więc zainstalowany jeden ze sterowników: Novell Synchronous/V.35+, Nowell Synchronous/RS-422+ lub Novell Synchronous/RS-232+.

NetWare Link/T1

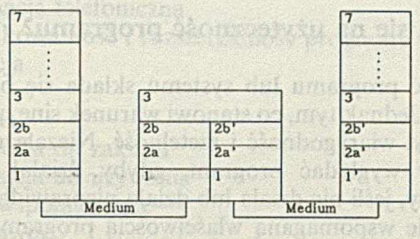
Podobnie jak w poprzednim przypadku produkt ten umożliwia łączenie sieci przez linię synchroniczną. Różnica polega na szybkości transmisji. W tym przypadku wynosi ona od 9600 b/s do 2,048 Mb/s.

Oprócz przedstawionych powyżej rozwiązań istnieją również produkty innych firm, realizujące łączenie sieci NetWare. Przykładem może być oprogramowanie i sterowniki firmy Gateway Communications: G/RemoteBridge64 i G/Remote Bridge. Umożliwiają one połączenie z odpowiednio 128 lub 64 zdalnymi sieciami lokalnymi przez linie transmisji danych, sieci z komutacją pakietów. Innym ciekawym rozwiązaniem są sterowniki MLB firmy Microcom, umożliwiające łączenie sieci za pomocą protokołu HLDC, sieci X.25 ora sieci ISDN. Umożliwiają one zastosowanie specjalnego oprogramowania firmy Microcom o nazwie Microcom Management Station, które pozwala zarządzać wszystkimi sterownikami. Istnieje również możliwość użycia protokołu SNMP (ang. *Simple*

Network Management Protocol) w celu zarządzania mostem międzysieciowym.

Zastosowanie węzłów międzysieciowych

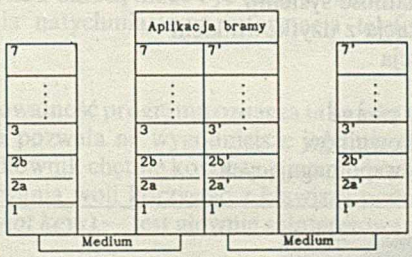
Architekturę węzła przedstawiono na rys. 4. Funkcje realizowane przez węzeł międzysieciowy obejmują: zmianę struktury adresów i rozmiarów pakietów oraz ewentualnie wyrównanie różnych poziomów usług łączących sieci. Istniejące na rynku realizacje sprzętowe węzłów międzysieciowych bardzo często zawierają w sobie funkcje mostów. Przykładem tego są produkty firmy 3Com: BR/2000 i BR/3000, umożliwiające realizację funkcji mostów i węzłów dla sieci z protokołami XNS, TCP/IP, OSI. Przykładem nowoczesnego węzła międzysieciowego jest ACS+ firmy Cisco System Inc. [2]. Umożliwia on łączenie sieci z protokołami TCP/IP, DECnet Phases IV i V, OSI, Apollo Domain, AppleTalk I i II oraz Banyan Vines. Połączenia te mogą być realizowane między sieciami Ethernet, TokenRing, FDDI oraz przez linie szeregowe o szybkości transmisji od 9,6 Kb/s do 4 Mb/s.



Rys. 4. Architektura łączenia LSK za pomocą węzła międzysieciowego

Zastosowanie bram międzysieciowych

Architekturę bramy międzysieciowej przedstawiono na rys. 5. Bramy stosuje się w celu połączenia dowolnego stanowiska roboczego z komputerami IBM pracującymi w sieci SNA, komputerami w sieci X.25 i innymi komputerami ze sprzęgiem asynchronicznym RS 232C. Zastosowanie bramy może również dotyczyć współpracy oprogramowania usługowego oferującego usługi sieciowe, np. poczty elektronicznej (3 + OpenReach/ /X.400, Soft-Switch SNADS Gateway/3 + Mail). Poniżej przedstawiono produkty realizujące funkcje bram międzysieciowych.



Rys. 5. Architektura łączenia LSK za pomocą bramy międzysieciowej

NetWare SNA Gateway ELS

Oprogramowanie to umożliwia zestawianie szesnastu połączeń z komputerami IBM w sieci SNA. SNA Gateway ELS w komunikacji ze stanowiskami roboczymi korzysta z protokołów Net-Ware SPX lub IBM NetBIOS. W stanowiskach roboczych jest wymagane oprogramowanie NetWAare 3270 LAN Workstation umożliwiające emulację terminala 3270 i drukarki 3270 oraz transmisję plików. Należy ono do standardowego wyposażenia systemu Advanced NetWare v.2.15 i SFT NetWare v.2.15.

NetWare SNA Gateway

Jest to produkt zgodny z omówionym powyżej, ale zapewniający większą liczbę obsługiwanych połączeń (do stu dwudziestu ośmiu). Dla przedstawionych produktów oferuje się również oprogramowanie umożliwiające zarządzanie, tj. monitorowanie, sterowanie i konfigurowanie połączeniami z dużymi komputerami. Program NetWare SNA Gateway Status Utility umożliwia zbieranie informacji o działaniu SNA Gateway oraz wszystkich współpracujących z nim stanowisk roboczych w sposób ciągły, z uaktualnianiem co kilka sekund. Istnieje również oprogramowanie diagnostyczne NetWare SNA Gateway Diagnostics Utilities, które pozwala przeprowadzić diagnostykę sterownika bramy SNA, sprawdzić połączenia bramy z siecią SNA oraz połączenia bramy ze stanowiskami roboczymi.

NetWare LU6.2

Jest to oprogramowanie umożliwiające użytkownikom tworzenie zastosowań zgodnych z SAA (ang. *Systems Application Architecture*), oraz dostęp do LU6.2 komputera IBM w sieci SNA. Oprogramowanie to jest wykonywane w komputerze pracującym jako brama międzysieciowa SNA.

NetWare 5250 Gateway

Oprogramowanie to uruchamiane na stanowisku roboczym zamienia je w stanowisko usługowe umożliwiające połączenie z minikomputerami IBM System/3x. Połączenie to realizuje się z użyciem protokołu SDLC. Dopuszcza się dziewięć jednoczesnych sesji terminali i (lub) drukarek (pięć dla jednego stanowiska). W stanowiskach roboczych jest wymagane oprogramowanie NetWare 5250 LAN Workstation. W oprogramowaniu NetWare 5250 Gateway do komunikacji komputer-stanowisko robocze stosuje się styk NetBIOS.

NetWare Asynchronous Communication Server (NACS)

Oprogramowanie to jest uruchamiane w stanowisku roboczym i umożliwia zestawienie szesnastu jednoczesnych połączeń asynchronicznych. Stanowisko to staje się stanowiskiem usługowym. Usługa dostępu do transmisji asynchronicznej nosi nazwę Distributed Asynchronous Name Service. Użytkownicy w stanowiskach roboczych muszą dysponować oprogramowaniem emulującym terminale, np. pakietem ASCOM IV.

NetWare X.25 Gateway

Oprogramowanie to umożliwia emulację terminali asynchronicznych przez sieć X.25. Brama jest połączona z siecią X.25 za pomocą modemu synchronicznego. Oprogramowanie zamienia stanowisko robocze w stanowisko usługowe X.25 Gateway, zapewniając dostęp do maszyn i publicznych baz danych. Użytkownicy dysponują oprogramowaniem emulacji terminali (oferuje się wiele różnych emulatorów). Zależnie od użytego emulatora można połączyć się maksymalnie z ośmioma komputerami jednocześnie. Produkt ten jest sprzedawany w dwóch wersjach: X.25 Gateway — umożliwia nawiązanie do trzydziestu dwóch sesji, dla maksymalnie ośmiu użytkowników, oraz X.25 Gateway Extended — do dwustu pięćdziesięciu czterech sesji dla dziewięćdziesięciu użytkowników. Rozwiązanie to wymaga pojedynczego połączenia z siecią publiczną, z użyciem karty NetWare X.25 Adapter (NetWare X.25 Extended Adapter).

Również firma Gateway Communications oferuje użytkownikom sieci NetWare rozwiązanie umożliwiające pracę z siecią X.25. Są to produkty o nazwie G/X.25 Gateway i G/X.25 Gateway & Bridge, które umożliwiają dostęp do zdalnych komputerów przez emulację terminali. G/X.25 Gateway zawiera 27 różnych emulatorów. Podobne funkcje realizuje produkt Maxess SNA Gateway firmy 3Com w stosunku do komputerów IBM w sieci SNA.

Dołączanie zdalnych stanowisk roboczych

Niektóre z sieci lokalnych umożliwiają przyłączenie zdalnych stanowisk roboczych. W przypadku sieci NetWare są możliwe dwa tryby współpracy zdalnych stanowisk z siecią. W pierwszym z nich stanowisko zdalne można przyłączyć do sieci lokalnej wykorzystując funkcje mostu zewnętrznego — NetWare Link/Asynch. Sprzęg MOC1/COM2 wraz z linią asynchroniczną są traktowane jak specyficzny sterownik sieciowy. Oprogramowanie uruchamiane przez użytkownika jest wykonywane w zdalnym stanowisku roboczym. W drugim przypadku zdalne stanowiska robocze przyłącza się do sieci za pośrednictwem asynchronicznego stanowiska usługowego (NACS). Każdemu zdalnemu stanowisku odpowiada określone stanowisko robocze w sieci, na którym jest wykonywane oprogramowanie NetWare. Zdalne stanowisko robocze jest traktowane jako nieinteligentny terminal, co oznacza, że wykonywane przez użytkownika przetwarzanie danych odbywa się poza zdalnym stanowiskiem roboczym (w odpowiadającym mu stanowisku w sieci lokalnej). W innym z rozwiązań tego trybu pracy zastępcze stanowiska robocze w sieci mogą być zastąpione dedykowanym stanowiskiem usługowym wykonującym oprogramowanie NetWare Access Server.

* * *

Na podstawie omówionych przykładów można stwierdzić, że funkcje łączenia lokalnych sieci komputerowych stanowią integralny ich składnik. Duża różnorodność możliwych rozwiązań wymaga szczegółowej ich oceny pod kątem wpływu na efektywność działania sieci.

W wyniku połączenia sieci komputerowych powstaje złożony system sieci, stwarzający liczne problemy w sferze zarządzania taką siecią. Każda z sieci ma bowiem własny system zarządzania. Systemy zarządzania LSK, w większości tych sieci, są systemami funkcjonalnie ograniczonymi. Ponadto urządzenia międzysieciowe od różnych dostawców są zarządzane według specyficznych protokołów tych firm, tworząc często niezależny system zarządzania. Przykładowo, w omawianej w artykule gamie produktów międzysieciowych tylko w niewielkiej ich liczbie jest implementowany protokół zarządzania SNMP będący standardem przemysłowym, a w żadnym z nich nie implementuje się protokołów zarządzania ISO-OSI, które są standardem międzynarodowym. Z tego względu zbudowanie zintegrowanego systemu zarządzania połączonymi sieciami jest bardzo trudne. Powoduje to z kolei nieciągłość funkcji zarządzania i prowadzi do opóźnień w międzysieciowym przesyłaniu danych oraz zmniejszenia pewności połączonych sieci. Nie jest to argument przeciwko łączeniu sieci komputerowych, lecz za koniecznością prowadzenia prac w kierunku rozwoju metod i środków do budowania zintegrowanych systemów zarządzania SK.

Z powyższego wniosku można też wyciągnąć wskazówkę praktyczną o potrzebie uwzględniania, przy wyborze urządzeń międzysieciowych, stopnia realizacji funkcji zarządzających i ich zgodności z systemami zarządzania łączonych sieci komputerowych.

dokończenie na s. 23

Artykuł zawiera opis głównych elementów składających się na użyteczność (ang. *usability*) programu. Omawia metody pracy projektanta pozwalające na uzyskanie bardzo dobrej jakości programu. Większość omawianych zagadnień zaczerpnięto z artykułu — J.D. Gould „How to Design Usable System".

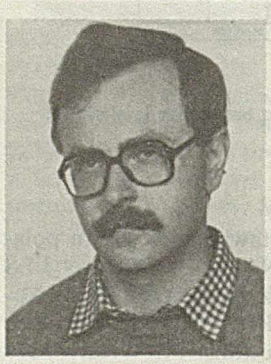
Co składa się na użyteczność programu?

Na jakość programu lub systemu składa się bardzo wiele czynników. Jednak tym, co stanowi warunek sine qua non jego istnienia, jest wiarygodność i rzetelność. Niezależnie od tego, jak mógłby wyglądać program, gdyby działał — jest on bezużyteczny, jeśli nie działa lub działa nieprawidłowo. Drugą bezwzględnie wspomaganą właściwością programu jest szybkość i jakość odpowiedzi na czynności wykonywane przez użytkownika. Brak odpowiedzi może doprowadzić nawet do zniszczenia sprzętu. W zaprzyjaźnionej firmie jedna z opcji programu była uruchamiana przy zwolnieniu klawisza — oczywiście znalazł się użytkownik, który męskim naciśnięciem niby nie kontaktującego klawisza zepsuł klawiaturę.

Badania wskazują, że szybkość odpowiedzi stanowi element najbardziej wpływający na zadowolenie użytkowników. Oczekiwanie powyżej 1 s może powodować zmniejszenie efektywności pracy o 40 i więcej procent.

Oto zestawienie wszystkich elementów wpływających na użyteczność:

- Działanie systemu
  - rzetelność
  - sprzężenie zwrotne
- Funkcjonalność systemu
- Komunikacja z użytkownikiem
  - organizacja
  - sprzęt
- Materiały pisane
  - dla użytkowników
  - dla osób wspomagających



Mgr inż. Maciej Andrzej Wierzba ukończył w 1980 r. Wydział Elektroniki Politechniki Warszawskiej (specjalność „Automatyka"), a w 1984 r. studia na Uniwersytecie Warszawskim — Wydział Matematyki, Informatyki i Mechaniki — o specjalności „Analiza Numeryczna". W 1986 r. rozpoczął pracę w Instytucie Informatyki Uniwersytetu Warszawskiego. Od 1988 r. kieruje zespołem wykonującym system organizacji komunikacji z użytkownikiem ViDE — 2.

ANDRZEJ MACIEJ WIERZBA
Instytut Informatyki
Uniwersytet Warszawski

Jak pisać programy satysfakcjonujące użytkownika?

- Tłumaczenie tekstów
— w programie
— w materiałach
- Wspomaganie nauki programu
— kursy dla użytkowników,
— system informacji na bieżąco w programie
— informacja telefoniczna
- Modyfikowalność i rozszerzalność programu
- Instalacja
- Serwis
- Reklama
— motywacja do zakupu
— motywacja do używania
- Osoby wspomagające
— sprzedawcy
— osoby szkolące użytkowników
— serwis
— systemowcy.

Na funkcjonalność systemu składają się operacje możliwe do wykonania przez użytkownika. Zbyt mały zestaw tych operacji może powodować, że system staje się bezużyteczny. Dołączenie operacji nigdy nie wykorzystywanych powoduje zaciemnienie struktury programu i utrudnienie w korzystaniu z niego.

Powstające oprogramowanie jest coraz większe i coraz bardziej skomplikowane. Uczenie jego używania nie może ograniczać się do wręczenia podręcznika. Użytkownicy dużo chętniej uczą się obserwując, jak pracują inni i zadając pytania. Podręcznik zawiera bardzo wiele informacji — wyłowienie tych, które są potrzebne na bieżąco, stanowi dla początkującego użytkownika duży problem. Pomocą może być także kontekstowy bryk oraz bezpośrednia natychmiastowa informacja telefoniczna (ang. *hot-line*).

Modyfikowalność programu oznacza taką jego konfigurowalność, która pozwala na wygodniejsze jego używanie. Początkujący użytkownik chętnie korzysta z menu i bryku; po długim okresie używania woli korzystać z klawiszy przyspieszających pracę (ang. *hot keys*) — jest głównie zainteresowany szybkością pracy.

Projektując program trzeba zastanowić się nad motywacją, dlaczego użytkownik kupi i będzie używał właśnie nasz program. Jeśli będzie on mniej funkcjonalny lub droższy, niż już istniejące, to mamy małą szansę na jego sprzedaż.

Materiały i kursy należy zapewnić nie tylko użytkownikom, ale także osobom związanym ze sprzedażą i serwisem. Typowym obrazkiem, który można obejrzeć na targach komputerowych, jest sprzedawca wymigujący się od demonstracji programu. W naszym interesie jest zapewnienie mu odpowiedniego kursu — mając do wyboru parę porównywalnych produktów, będzie sprzedawał właśnie nasz.

Reguły projektowania systemów

Podane elementy użyteczności nie odpowiadają na pytania:
— Jak robić takie systemy?

— Jak połączyć wszystkie elementy użyteczności w całość?

Badania nad tymi problemami doprowadziły do znalezienia czterech reguł projektowania systemów:

- wczesne i ciągłe skupianie się na użytkowniku, polegające na wywiadach, współpracy, obserwacjach, analizie, wizytach w miejscu pracy;

- projektowanie zintegrowane — wspólne branie pod uwagę wszystkich czynników użyteczności;

- jak najwcześniejsze i ciągłe testowanie makiet, prototypów i programu w rzeczywistych warunkach;

- projektowanie iteracyjne, polegające na ciągłych modyfikacjach wynikających z testowania.

Ciągłe skupianie się na potrzebach i wymaganiach użytkownika.

Zadaniem projektanta jest danie użytkownikowi takiego systemu, dzięki któremu będzie mógł pracować lepiej i przyjemniej. Nie można bez konsultacji z użytkownikiem zadecydować, czego on potrzebuje, co lubi, co i jak może wykonywać.

Pierwszym krokiem jest dokładne określenie, **kim będzie użytkownik**. Dość często nie będzie on programistą, nie będzie wiedział, co to są liczby szesnastkowe, nie będzie rozumiał, co to jest system operacyjny (spora grupa programistów zakłada, że takie informacje są ogólnie znane). Typowym błędem jest projektowanie systemów skomplikowanych (dających olbrzymie możliwości) na tyle, że może z nich korzystać tylko twórca. Nawet jeśli biuro znajduje się w tym samym budynku, programista zawsze wie lepiej i nie zadaje sobie trudu na zapytanie urzędnika, co jest mu potrzebne. Czasem bierze się to także z tego, że programista uważa swoją pracę za nowatorską i wiekopomną — nie bierze jednak pod uwagę możliwości przyzwyczajeń użytkownika.

Niezależnie od tego, jak dobrze zostanie opisany użytkownik, musimy zdać sobie sprawę, że jest to wyłącznie jego **przybliżenie** — prawdziwą twarz ukaże on dopiero podczas testowania programu.

Metody poznania użytkownika

Rozmowa z użytkownikiem

Rozmowa z użytkownikiem pozwala dowiedzieć się, co i jak on robi, jakie ma problemy i co chciałby zmienić. Odpowiada, czy jest on zainteresowany zmianą na swoim stanowisku pracy. Bardzo często użytkownik jest przerażony komputeryzacją, obawia się, czy da sobie radę w nowych warunkach — fakt ten trzeba uwzględnić przy projektowaniu systemu. Korzystne jest przeprowadzenie kursu wstępnego tłumaczącego personelowi, dlaczego będzie im się pracowało lepiej i uzmysławiającego łatwość obsługi. Umożliwia to lepszą współpracę z wykonawcami systemu.

Nie należy oczekiwać, że nasi rozmówcy wymyślą system, bezwzględnie muszą oni jednak już w fazie projektowej zatwierdzać nasze pomysły. Stałe spotkania dotyczące projektu powinny odbywać się w miejscu zainstalowania systemu. Dobrym pomysłem jest przeniesienie tam zespołu opracowującego. Dużo lepiej jest utrzymywać bezpośredni kontakt z użytkownikiem niż zlecać dokonanie ekspertyzy, np. specjalistom od marketingu. To, co opiszą oni w swoich opracowaniach, będzie tylko w pewnym stopniu odpowiedzią na nasze pytania. Jeśli projektowalibyśmy system dla Marsjan, na pewno chcielibyśmy się dowiedzieć, ile mają rąk, jakie oczy, czy słyszą i jaką mają zdolność pojmowania. Nie popełnimy żadnego błędu traktując naszych klientów właśnie w ten sposób.

Wizyta w miejscu instalowania systemu

Wybór konkretnego systemu zależy od miejsca jego zainstalowania. Inne wymagania są stawiane w przypadku instalacji w hali fabrycznej, a inne dla terminala zainstalowanego w cichym pokoju. O ile w pierwszym przypadku nacisk należy położyć na pancerną obudowę, o tyle w drugim o jakości będzie decydował cichy wentylator w termowyłącznikiem.

Obserwacja pracy użytkownika

W wielu miejscach istnieją już zainstalowane systemy, z których użytkownicy są mniej lub bardziej zadowoleni. Obserwacja pracy systemu może doprowadzić do wielu ciekawych spostrzeżeń:
— sekretarki bardzo chętnie korzystają z menu, ale boją się wypisywać polecenia;
— urzędnik ma zawsze za mało miejsca na biurku;
— użytkownik niechętnie używa narzędzi, na których uruchomienie musi dłużej czekać;
— ludzie sięgają po opisy tylko w ostateczności.

Wszystkie spostrzeżenia należy uwzględnić w nowym systemie. Przykładowo, zamiast zmuszać użytkownika do sięgania po podręcznik, dużo lepiej jest zrobić dobry kontekstowy bryk.

Rejestracja pracy użytkownika na kasecie video

Użytkownika przy pracy z istniejącym systemem można zarejestrować na kasecie wideo. Wybrane momenty, w których ma on trudności z obsługą systemu lub wyraża niezadowolenie (np. zbyt długi dostęp do bazy) należy pokazać wszystkim członkom zespołu projektantów.

Zapoznanie się z systemem organizacji

Tworzone oprogramowanie musi być zgodne z systemem organizacji przedsiębiorstwa. Jeśli organizacja pracy jest wątpliwa, to przed przystąpieniem do komputeryzacji trzeba ją bezwzględnie poprawić.

Namówienie użytkownika, aby głośno myślał

Dobrą metodą poznania sposobu myślenia użytkownika jest poproszenie go, by głośno komentował to, co robi podczas pracy z programem. Obserwacje takie pozwalają usunąć niejasne, trudne dla użytkownika fragmenty.

Próba samodzielnej pracy

Warto czasem siąść przed monitorem i zacząć pracować jak urzędnik, korzystając z jego rad jak wykonać poszczególne operacje. Możemy w ten sposób znaleźć najczęściej wykonywane sekwencje i starać się je zoptymalizować w projekcie.

Włączenie do projektowania użytkowników

Takie postępowanie może być szczególnie użyteczne, jeśli jest budowany system pod klucz — osoby pracujące w przedsiębiorstwie najlepiej wiedzą, co jest w nim potrzebne.

Wykorzystywanie ekspertów

Do pewnych aspektów należy zatrudniać ekspertów mających duże doświadczenie i długą praktykę.

Analiza zadania

W wielu przypadkach jest konieczna formalna analiza zadania. Może być ona wykonywana np. przy całkowicie nowej konstrukcji, wymagającej naukowego przebadania układu człowiek — system.

Ankiety

Informacje uzyskane podczas wywiadów pomagają w przygotowaniu ankiet, które powinni wypełnić wszyscy użytkownicy systemu.

Stworzenie testów jakości systemu i ustalenie limitów

Z technicznego punktu widzenia jest możliwe ustalenie rozdzielczości ekranu lub szybkości pracy. Warto także ustalić wymagania co do zachowania systemu, np. 90% urzędników w przedsiębiorstwie powinno po jednodniowym szkoleniu wprowadzać do bazy wyspecyfikowane dane o jednej osobie w czasie krótszym niż 4 minuty.

Projektowanie zintegrowane

Elementy składające się na użyteczność programu nie mogą być rozważane rozłącznie — są one tak powiązane, że rozdzielenie ich doprowadziłoby do pogorszenia jakości programu. Główne problemy projektowania zintegrowanego mają charakter organizacyjny. Nie można do projektowania systemu zatrudnić członków wyspecjalizowanych działów firmy, zajmujących się wyłącznie podręcznikami lub szkolących obsługę serwisową. Dla każdego projektu jest konieczne stworzenie oddzielnego zespołu zajmującego się wszystkimi aspektami użyteczności jednocześnie. Zespół ten powinien mieć zapewnioną odpowiednią kadrę, pieniądze, czas i uprawnienia. Jego szef jest osobą odpowiedzialną za jakość powstającego systemu.

Ciągłe testowanie systemu

Przedsiębiorstwo zamawiające system chce go otrzymać i używać możliwie jak najszybciej. Nie ma czasu na roczną fazę uruchomieniową, podczas której zostaną poprawione wszystkie błędy i uwzględnione zalecenia użytkowników. Wynikiem tego jest wymaganie jak naj-

szybszego przekazywania napisanych fragmentów projektu lub programu do testowania przez użytkownika. Im wcześniej są robione zmiany w programie, tym mniejszy jest ich koszt.

Stworzenie scenariusza

Chcąc rozmawiać z użytkownikiem musimy zbliżyć się do świata jego pojęć. Nie zainteresujemy go złożonością obliczeniową ani wyszukanymi algorytmami. Najwygodniejszym językiem są obrazki na ekranie, klawisze myszy i klawiatury. Użytkownik chce wiedzieć, po zobaczeniu jakiego obrazka ma nacisnąć np. klawisz *Enter*, aby wykonała się żądana operacja. Temu celowi ma służyć wykonanie scenariusza składającego się z rysunków lub nagranie takiego zestawu rysunków i czynności na magnetowid. To drugie jest szczególnie ważne w przypadku stosowania dużej ilości grafiki i animacji.

Jednoczesne pisanie podręcznika

Podręcznik powinien być pisany równocześnie z projektem systemu — dużo wcześniej niż zostanie napisany pierwszy wiersz programu. Oprócz tekstu powinny w nim znaleźć się rysunki ze scenariusza. Użytkownicy powinni zadecydować, czy jego zawartość ich satysfakcjonuje. Jeśli okaże się, że nie, to musi być on napisany od nowa. Odpowiedzi na pytania zadane przez użytkowników należy dołączyć do nowej wersji.

Tworzenie prototypu

W przypadku używania w pracy gotowego systemu organizacji komunikacji z użytkownikiem jest możliwe **tanie stworzenie jego prototypu** bez implementowania części funkcjonalnej programu. Stworzenie prototypu zmienia jakość pracy — stymuluje dyskusje i stwarza lepszą motywację do pracy (to już działa, nie trzeba pisać od nowa, tylko dopisywać).

Demonstrowanie programu

Nawet zespół doświadczonych projektantów nie jest w stanie przewidzieć wszystkich szczegółów systemu. Jego prezentacja i uwagi osób spoza zespołu stwarzają możliwość wprowadzenia poprawek w możliwie wczesnej fazie projektu.

Obserwacja użytkownika

Prototypy i kolejne wersje programu oraz podręczniki powinny być testowane przez użytkowników. Informacje o koniecznych zmianach mogą być otrzymywane od użytkownika bezpośrednio przez wspólną pracę, oraz przez obserwację jego pracy na nagranej taśmie wideo.

Publikacje

Informacje o powstającym programie powinny być udostępniane możliwie szerokiemu gronu, z intencją otrzymania możliwie wielu krytycznych uwag.

Oficjalne testowanie programu

Poza osobami okazjonalnie testującymi program bez żadnej większej motywacji, powinien on być przetestowany przez zespół, któremu finansowo bądź ambicjonalnie będzie zależało na znalezieniu możliwie dużej liczby błędów.

Projektowanie iteracyjne

Praca nad systemem wymaga ciągłego uwzględnienia zmian. Wprowadzenie każdej zmiany obejmuje następujące kroki:

- stwierdzenie potrzeby zmiany,
- stwierdzenie możliwości realizacji,
- chęć wprowadzenia,
- dokonanie zmiany w systemie.

Im bardziej zaawansowane są prace, im większy jest zespół i im słabsza jest dokumentacja, tym zmiana jest kosztowniejsza. Aby mieć możliwość zmian, zespół powinien dysponować odpowiednimi narzędziami. Na pewno do tworzenia dialogu z użytkownikiem powinien on używać systemu organizacji komunikacji z użytkownikiem. Pozwala to oddzielić część operacyjną od części zarządzającej dialogiem, dzięki czemu łatwo można tworzyć część prototypu, różne wersje językowe oraz konkretny bryk. Dużym ułatwieniem przy pisaniu i pielęgnowaniu oprogramowania mogą być systemy CASE.

Należy liczyć się z faktem, że poszczególni członkowie zespołu będą starali się ograniczyć liczbę zmian, zwłaszcza w swoich fragmentach. Praca programisty polega także na robieniu zmian; udawanie, że są one niepotrzebne, świadczy o jego słabych kwalifikacjach.

Reguły i praktyka

Skoro podane reguły są tak jasne i proste oraz gwarantują dobrą jakość produktu, to dlaczego nie są one powszechnie stosowane?

Główne przyczyny tkwią w trudnościach organizacyjnych i motywacyjnych:

- w tradycyjnej strukturze firmy jest przesuwać pracowników do coraz to nowych grup projektowych;
- kontrola dynamicznie zmieniającego się systemu jest bardzo trudna i często przerasta możliwości intelektualne szefa projektu,
- uczestnicy projektu dla własnej wygody (szybkości pracy) wolą skupiać się tylko na zleconym im fragmencie;
- wprowadzanie zmian wymaga dobrego kosztownego udokumentowania pisanych od nowa fragmentów kodu;
- kontakt z użytkownikiem nie sprawia przyjemności programiście zaopatrzonemu głównie w optymalizację algorytmów.

Należy mieć nadzieję, że te zespoły programistów, które zamierzają zdobyć znaczącą pozycję na rynku informatycznym, wkrótce dostosują się do opisanych zasad. Pociągnie to za sobą zmniejszenie strat wynikających z błędów popełnionych w fazie projektowania systemów.

LITERATURA

[1] Handbook of Human-Computer Interaction. M. Helander (Ed.), North-Holland 1988.

Łączenie lokalnych sieci...

dokończenie ze s. 20

LITERATURA

[1] Berntsen J. A., Davin J. R., Pitt D. A., Sullivan N. G.: MAC Layer Interconnection of IEEE 802 Local Area Networks. Computer Networks and ISDN Systems, Vol. 10, 1985

[2] Brander S. O.: Testing Multiprotocol Routers: How Fast is Fast Enough? Data Communications International, Febraury 1991

[3] International Standard Organization — Information processing systems — Data communications — Protocol for prociding the connectionless-mode network service

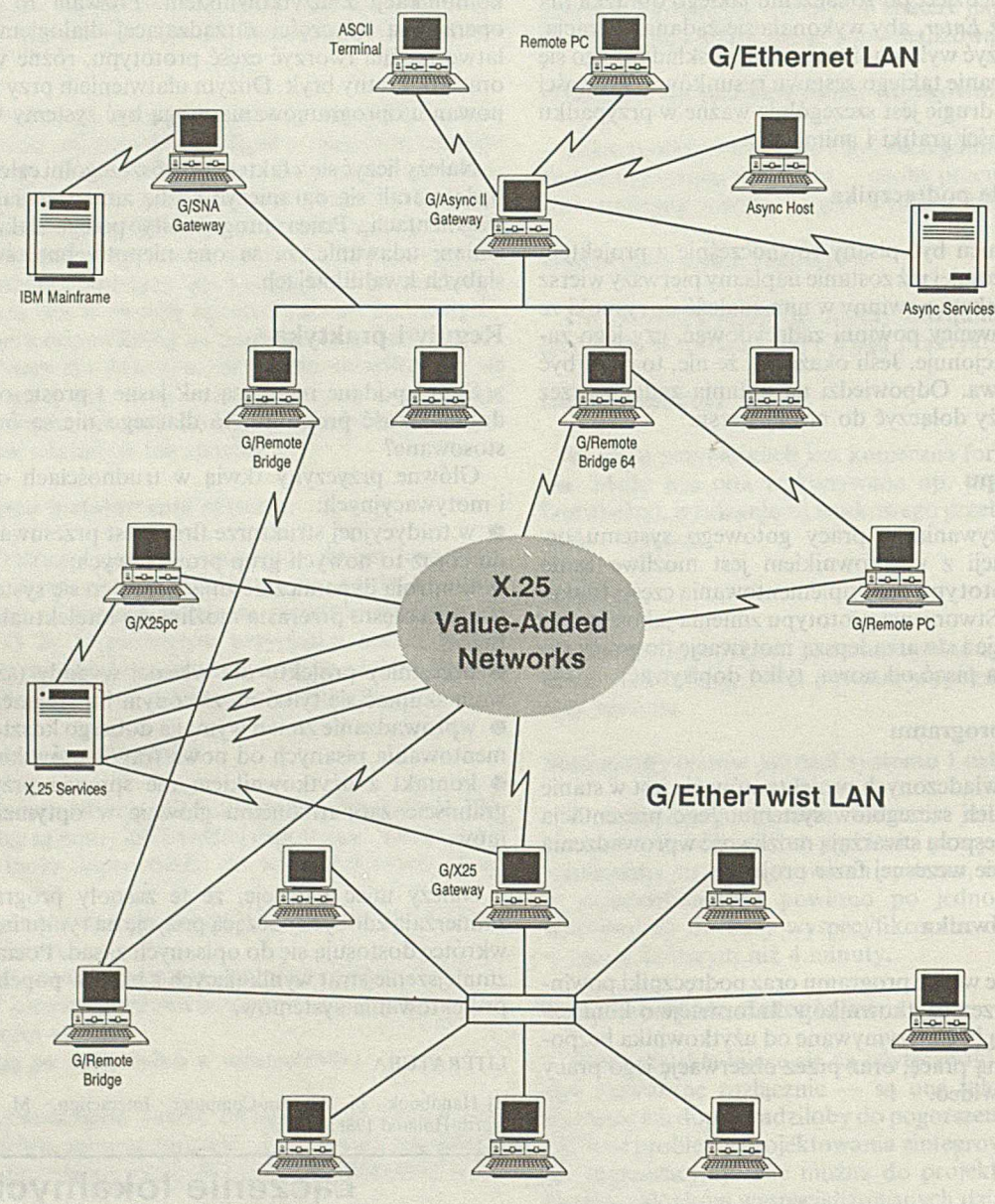
[4] Internet Protocol DARPA Internet Program Protocol Specification, September 1981, RFC791

[5] Lamb Ch.: The Use of Bridges in Local Area Networks. Proc. of 13 th Conf. on LAN, Minneapolis, Minesota 1988

[6] Stallings W.: When One LAN Is Not Enough. Byte, January 1989.

Produkty sieciowe LAN i WAN

Gateway communications, inc.



BEZKONKURENCYJNE PRODUKTY
SIECI KOMPUTEROWYCH NAGRADZANE PRZEZ:

PC MAGAZINE, LAN MAGAZINE, INFO WORLD

OFERUJE AUTORYZOWANY DYSTRYBUTOR:

MIKROB

P.W.P.T. MIKROB SP. Z O.O.

20-346 Lublin, ul. Długa 5

Tel. (0-81) 422-28, faks 415-43, teleks 643776 MIKRO PL

Na życzenie wysyłamy katalog produktów. Atrakcyjny program współpracy dla dealer'ów.

O/7/91

MARIA BRONOWSKA
Instytut Podstaw Informatyki PAN
Centrum Obliczeniowe
Warszawa

Procedury graficzne systemu RAMTEK RM-9460

Konfiguracja sprzętowa i możliwości systemu RAMTEK, zainstalowanego przed trzema laty w Centrum Obliczeniowym IPI PAN w Warszawie, obejmująca minikomputer Delta 8000 typu VAX/750 oraz system graficzny RAMTEK RM-9460 wraz z funkcjami sprzęgu (sprzętowymi i programowymi), w wersji dostarczonej przez producenta [4], zostały szczegółowo opisane w [2]. Sprzęg ten umożliwia m.in. wysyłanie z komputera głównego do RAMTEK-a tablic słów 16-bitowych, zawierających instrukcje graficzne, oraz odczytywanie tablic z RAMTEK-a przez wywołanie odpowiednich procedur w Fortranie. RAMTEK natomiast dekoduje i przetwarza zbiór ponad 150 binarnie sformatowanych instrukcji. Z tego ponad połowa może być modyfikowana przez jeden lub kilka argumentów (wybranych z 22 możliwych). Zapewnia to maksymalną elastyczność funkcjonalną instrukcji, ale utrudnia programowanie. Z tego m.in. powodu zaprojektowano i wykonano pakiet RAMTEK RM-9460, zawierający około 70 procedur graficznych PSG-podobnych [1, 3], które zdecydowanie ułatwiają programowanie grafiki w systemie RAMTEK [5].

Procedury (napisane w Fortranie) są tak dobrane, by tworzyły spójny pakiet i udostępniały w łatwej do stosowania postaci jak największą liczbę instrukcji systemu, bez utraty ich elastyczności.

Pracę systemu graficznego inicjuje (i nadaje indeks CHAN stacji graficznej) procedura pomocnicza INIGR. Inna procedura pomocnicza COLOUR pozwala wybrać paletę 8, 16, 64 lub 256 kolorów, 256 odcieni szarości oraz „kolory tęczy". Jest to możliwość wstrzymania wykonywania procedur graficznych na określony (w sekundach) czas procedurą WAIT (tab. 1).

Tabela 1. Procedury pomocnicze

Lp.	Nazwa	Instrukcja RAMTEK-a	Funkcja
1	BYE	RMDSIN	Zakończenie pracy systemu
2	COLOUR	—	Wczytanie palety kolorów
3	INIGR	RMASIN	Zainicjowanie pracy systemu
4	LOADMEM	03	Definiowanie koloru
5	SHCOL	—	Wyświetlanie palety
6	SETPASS	—	Ustalenie reakcji na błąd
7	WAIT	RMWAIT	Opóźnienie w sekundach
8	WAITR	29	Opóźnienie w cyklach ekranu

W pakiecie procedur zachowano koncepcję oprogramowania firmowego RAMTEK-a przekształcania obrazu do kolejnych układów współrzędnych (rys.).

układ współrzędnych lokalnych → układ współrzędnych globalnych → układ współrzędnych pamięci obrazu → układ współrzędnych ekranu.

Tabela 2. Procedury graficzne

Lp.	Nazwa	Instrukcja RAMTEK-a	Funkcja
1	ARC1	6E	Rysowanie łuku przez 3 punkty
2	ARC2	6F	Rysowanie łuku o kącie rozwarcia alfa
3	ARC3	70	Rysowanie łuku o promieniu *R* przez 2 punkty
4	CIRC	6D	Rysowanie okręgu
5	CONIC	OF	Rysowanie krzywej stożkowej
6	ERASE	09	Wypełnianie wielokąta kolorem
7	FILL	3A	Zmiana koloru obszaru
8	MARK	—	Rysowanie 9 rodzajów znaczników
9	PLOTBOX	10	Rysowanie wykresu słupkowego
10	PLOTLIN	37	Rysowanie wykresu liniowego
11	PLOTP	36	Rysowanie wykresu punktowego
12	POINT	38	Rysowanie punktu
13	POINS	38	Rysowanie wielu punktów
14	POLY	98	Wypełnianie wielokąta kolorem
15	RASTER	OD	Rysowanie obrazu w postaci rastrowej
16	TEXT	OC	Pisanie tekstu
17	VEC	OE	Rysowanie wektora
18	VECL	OE	Rysowanie łamanej
19	VECUNL	OF	Rysowanie wektorów

Wszystkie procedury graficzne (tab. 2) definiują obraz w lokalnym układzie współrzędnych. Parametr MODE pozwala natomiast stosować różne tryby adresowania. Do dyspozycji są cztery tryby:

- bezwzględny,
- indeksowy 1,

Tabela 3. Procedury przekształceń współrzędnych i operowania macierzami transformacji

Lp.	Nazwa	Instrukcja RAMTEK-a	Funkcja
1	INIM	57	Inicjowanie macierzy *NRM*
2	LOADM	53	Kopiowanie macierzy *NRM* do macierzy 0
3	MULM	55	Mnożenie dwóch macierzy *NRM1* i *NRM2*
4	MULM	56	Mnożenie macierzy *NRM* przez podaną
5	POPM	51	Zdejmowanie macierzy ze stosu
6	PUSHM	50	Odkładanie macierzy na stos
7	READCLIP	24	Odczytywanie prostokąta obcinania
8	READINDX	24	Odczytywanie wartości indeksu
9	READOFFSET	24	Odczytywanie położenia okna obcinania
10	READORIGIN	24	Odczytywanie położenia układu ekranu
11	ROTM	5A	Obracanie obrazu
12	SCALEM	58	Skalowanie obrazu
13	SCROLLX	12	Przesuwanie obrazu wzdłuż osi *X*
14	SCROLLY	11	Przesuwanie obrazu wzdłuż osi *X*
15	SELORIGIN	27	Ustalanie orientacji układu współrzędnych
16	SETCLIP	08	Ustalanie prostokąta obcinania
17	SETINDX	08	Ustalanie wartości indeksu
18	SETM	52	Wypełniania macierzy *NRM*
19	SETOFFSET	08	Ustalanie położenia prostokąta obcinania
20	SETORIGIN	08	Ustalanie położenia układu ekranu
21	STOREM	54	Kopiowanie macierzy 0 do macierzy *NRM*
22	TRANSM	59	Przesuwanie obrazu
23	ZOOM	26	Sprzętowe powiększanie obrazu

- indeksowy 2,
- przyrostowy.

System RAMTEK zapamiętuje osiem macierzy, ponumerowanych od 0 do 7. Macierz 0 jest macierzą transformacji. System zawiera również stos macierzy o ośmiu poziomach, na wierzch którego można odkładać macierz o zadanym numerze *NRM* lub zdejmować i przesyłać do macierzy *NRM*. Współrzędne przekazywane z komputera głównego VAK/750 do RAMTEK-a są przekształcane do układu współrzędnych globalnych za pomocą automatycznego sprzętowego mnożenia przez macierz transformacji i zapisywane w pamięci obrazu. Macierz tę należy ustalić przed użyciem procedur graficznych, korzystając z procedur operowania macierzami (tab. 3). Pozwalają one definiować macierze skalowania, obrotu, przesunięcia, zapamiętywać macierze w systemie RAMTEK, odkładać na stos i zdejmować z niego macierze, mnożyć przez siebie itd.

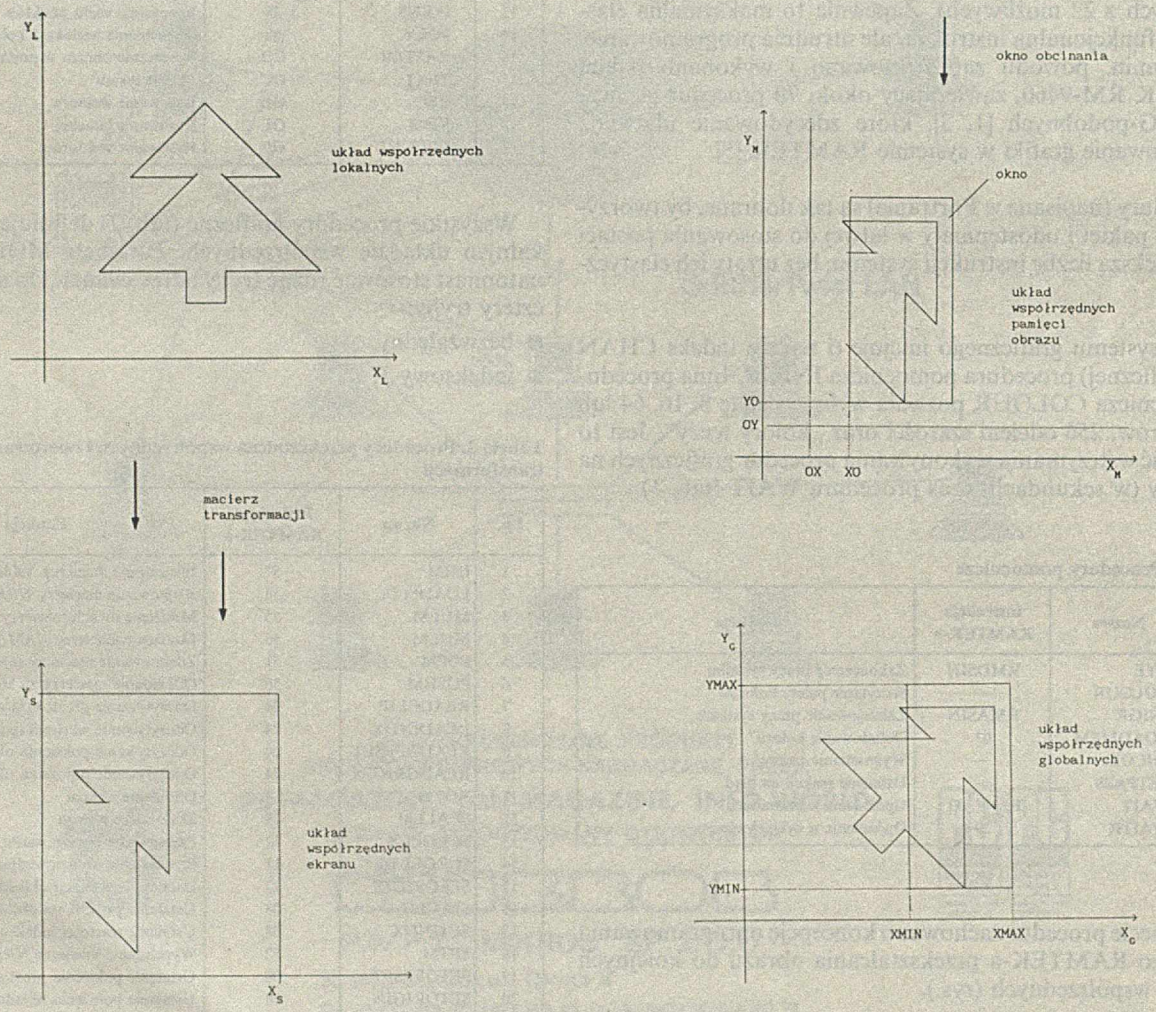
We współrzędnych globalnych jest ustalane okno obcinania, tj. prostokąt o bokach równoległych do osi, a we współrzędnych pamięci obrazu — położenie tego prostokąta.

Dane z pamięci obrazu są wyświetlane na monitorze graficznym RAKTEK-a, przy czym można określić położenie współrzędnych ekranu względem współrzędnych pamięci obrazu. Jest możliwe sprzętowe dokonanie zbliżenia (ang. *zoom*) lub przewijania (ang. *scroll*) na ekranie, zadawanego parametrami wywoływanej procedury.

W celu rozszerzenia możliwości procedur graficznych wprowadzono pewne parametry definiowane osobnymi procedurami (tab. 4). Istnieją również odpowiednie procedury odczytujące te parametry z systemu RAMTEK. Można zdefiniować np. wzór

Tabela 4. Procedury parametrów graficznych

Lp.	Nazwa	Instrukcja RAMTEK-a	Funkcja
1	READCOL	24	Odczytywanie koloru
2	READCOP	24	Odczytywanie punktu bieżącego
3	READPAT	24	Odczytywanie wzoru
4	READSCAN	24	Odczytywanie kierunku wypełniania *SCAN*
5	READSIZE	24	Odczytywanie współrzędnych powielania piksli
6	READSPAC	24	Odczytywanie odstępów między znakami
7	READWINDOW	24	Odczytywanie bieżącego okna
8	SETCOL	08	Ustalenie koloru
9	SETCOP	08	Ustalenie bieżącego punktu
10	SETDEFAULT	06	Ustawienie domyślnych wartości
11	SETPAT	08	Ustalenie wzoru wektora
12	SETSCAN	08	Ustalenie kierunku wypełniania
13	SETSIZE	08	Ustalanie współrzędnych powielania piksli
14	SETSPAC	08	Ustalenie odstępów między znakami
15	SETWINDOW	08	Ustalanie współrzędnych okna



Przekształcenie obrazu z lokalnego układu współrzędnych do układów współrzędnych ekranu

linii wektora, ustalanie koloru podstawowego i koloru tła, ustalanie rozmiaru i orientacji znaków, ustalanie kierunku pisania w wierszu, odległości między znakami w wierszu i między wierszami.

Pakiet zawiera rozbudowane mechanizmy wykrywania i obsługi błędów, wykorzystujące m.in. możliwości systemu operacyjnego VMS do tworzenia i wysyłania komentarzy na terminal użytkownika. Istnieje sposób ustalenia reakcji po wykryciu błędu: wyjście z procedury graficznej, w której wystąpił błąd programowania graficznego, lub wyjście z programu wraz z zakończeniem pracy systemu.

Na przykład, narysowanie okręgu o promieniu *R* i środku w punkcie *X*, *Y* kolorem czerwonym, obciętego oknem *XMIN*, *YMIN*, *XMAX*, *YMAX*, umieszczonym w punkcie *XW*, *YW* ekranu wymaga wysłania z komputera do RAMTEK-a ciągu firmowych instrukcji systemu w postaci tablicy 13 słów 16-bitowych. Nowy pakiet procedur umożliwia zastąpienie tej tablicy wywołaniem następujących trzech procedur:

CALL SETCLIP (CHAN, XMIN, YMIN, XMAX, YMAX, MODE)
CALL SETOFFSET (CHAN, XW, YW)
CALL CIRC (CHAN, X, Y, R, COL, MODE)

Pakiet procedur graficznych dla RAMTEK-a jest wygodnym narzędziem programowania grafiki i jest otwarty na dalsze uzupełnienia, wykorzystujące bardziej wyszukane możliwości tego systemu graficznego.

LITERATURA

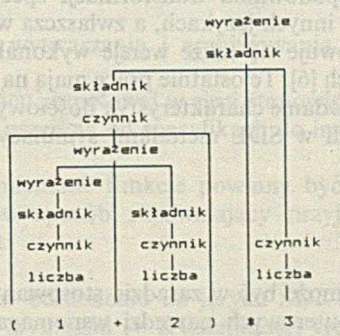
[1] Bronowska M., Mokrzycki W.: Oprogramowanie rastrowego monitora graficznego w systemie MERA 400. Archiwum opracowań IMM, Warszawa, 1981
[2] Bronowska M.: System graficzny RAMTEK RM-9460. Informatyka, nr 5, 1990
[3] Mokrzycki W.: System programowania grafiki komputera PSG 2.2. Informatyka, nr 3, 1987
[4] RM-9460 Graphic Display System Software Reference Manual. Ramtek Corp. No. 8000081-02A, July 1983
[5] System RAMGRAF. Opis użytkownika. Centrum Obliczeniowe IPI-PAN, Warszawa, grudzień 1990.

/unix

Programowanie w języku awk (2)

dokończenie ze s. 39

Przykładowo, **wyrażenie** jest sumą lub różnicą **składników**, a **składnik** iloczynem lub ilorazem **czynników**, co zapewnia prawidłową kolejność wykonania operacji. Drzewo wywodu dla wyrażenia (1 + 2) * 3 będzie miało postać:



Klasyczną drogą napisania kompilatora wyrażenia arytmetycznego będzie skonstruowanie analizatora składniowego. Jak to niżej będzie pokazane, napisanie gramatyki dla języka bardzo prosto przekłada się na program w języku **awk**. Każdemu symbolowi nieterminalnemu z gramatyki odpowiada funkcja w programie napisanym w **awk**: funkcja **wyrażenie** przetwarza składniki oddzielone znakami dodawania lub odejmowania, funkcja **składnik** przetwarza czynniki oddzielone znakami mnożenia lub dzielenia. Funkcja **czynnik** rozpoznaje liczby oraz wyrażenia zamknięte w nawiasy.

Niżej przedstawiony program pobiera wiersz pliku wejściowego jako pojedyncze wyrażenie arytmetyczne, po czym oblicza i wypisuje wartość wyrażenia. Liczba jest reprezentowana w postaci spotykanej w większości języków programowania: w postaci ciągu cyfr (liczba całkowita) lub w postaci dwóch ciągów cyfr oddzielonych kropką, przy czym wystąpić musi co najmniej jeden z ciągów cyfr (liczba zmiennoprzecinkowa). Jedynym, może nieco uciążliwym wymaganiem jest, aby wszystkie atomy leksykalne, a więc symbole operatorów i nawiasów oraz liczby były oddzielone spacjami.

```
NF > 0 {
    f = 1
    e = wyrazenie()
    if( f<=NF )
        printf("Blad w %s\n",$f)
    else
        printf("\t%.8g\n",e)
}

function wyrazenie( e ) {
    e = skladnik()
    while( $f=="+" || $f=="-" )
        e = $(f++) == "+" ? e+skladnik()
          : e-skladnik()
    return e
}


function skladnik( e ) {
    e = czynnik()
    while( $f=="*" || $f=="/" )
        e = $(f++) == "*" ? e*czynnik()
          : e/czynnik()
    return e
}

function czynnik( e ) {
    if( $f ~ /^[+-]?([0-9]+[.]?[0-9]*|[.][0-9]+)$/)
        return $(f++)
    else if( $f == "(" ) {
        f++
        e = wyrazenie()
        if( $(f++) != ")" )
            printf("blad: brak ) w %s\n",$f)
        return e
    }
    else {
        printf("blad: ani liczba ani ) w %s\n",$f)
        return 0
    }
}
```

Zmienna f wskazuje na tekst kolejnego atomu leksykalnego, którym jest liczba, symbol operatora lub symbol nawiasu. Na zmiennej e pamiętana jest wartość aktualnie obliczonej części wyrażenia. Warto zwrócić uwagę na fakt, że bądź co bądź złożona gramatyka **liczby** jest zdefiniowana w pierwszym wierszu funkcji **czynnik**.

Autor ma nadzieję, że zamieszczony wyżej przykład skłoni Czytelników do potraktowania języka **awk** jako naprawdę użytecznego i silnego narzędzia w pisaniu kompilatorów prostych języków programowania zwłaszcza wtedy, gdy dysponujemy bardzo małą ilością czasu. (ciąg dalszy nastąpi).

WŁADYSŁAW MAJERSKI

LITERATURA

[1] Aho A. V., Kernighan B. W., Weinberger P. J, The AWK Programming Language, Addison-Wesley, 1988.

PIOTR OSTROWSKI
MAREK ŚREDNIAWA
Instytut Telekomunikacji
Politechnika Warszawska

Język specyfikacji SDL (2)

W pierwszej części artykułu opisano język SDL, przedstawiając m.in. pojęcia podstawowe, model semantyczny i strukturę specyfikacji. Druga część zawiera omówienie problemów zastosowań, środowisk wspomagających (w tym system SPRITE opracowany w IT PW), a także tendencji rozwojowych.

Problemy zastosowań

Niedoskonałość specyfikacji, projektów i ich implementacji powoduje często nieoczekiwane, błędne zachowania systemu, zwłaszcza przy komunikacji procesów. Istotnym czynnikiem komplikującym problem jest losowość procesów transmisji (determinizm komunikacji) i losowe zjawiska występujące w urządzeniach (zawodność urządzeń i oprogramowania — niedeterminizm realizacji funkcji komponentów protokołu). Także pewne konstrukcje języka SDL są potencjalnie niebezpieczne w użyciu i uniemożliwiają sformułowanie reguł budowy poprawnej specyfikacji. Studia przeprowadzone w Instytucie Telekomunikacji Politechniki Warszawskiej doprowadziły do zaproponowania uściślonej wersji języka, nazwanej JS SDL [15], umożliwiającej budowę specyfikacji spełniających niektóre kryteria poprawności pragmatycznej [16, 20].

Opracowano reguły konstrukcji specyfikacji spełniających cztery następujące kryteria: **determinizmu (funkcjonalności), trwałości, odtwarzalności** i **spójności** (por. (17, 1, 13, 18)).

Problemy poprawności specyfikacji

Poprawność specyfikacji określa się dla poziomu pragmatycznego i odnosi do specyfikacji poprawnych syntaktycznie (zgodnych z definicją języka). Cechy poprawności pragmatycznej odnoszą się do zachowania modelu abstrakcyjnego systemu, określonego specyfikacją, ściślej — do interpretacji specyfikacji w modelu semantycznym. Stwarza to konieczność określenia dodatkowej cechy specyfikacji, wiążącej ją z zachowaniem się określonego nią modelu abstrakcyjnego — z właściwościami metaimplementacji. Cechę tę określano mianem reprezentatywności specyfikacji [17].

Okazuje się, że tworzenie reprezentatywnych definicji procesów nie jest prostym zadaniem. Definicje niereprezentatywne natomiast sugerują fałszywe interpretacje, utrudniają odczytanie specyfikacji, a w przypadku programów, prowadzą do nieoczekiwanych realizacji (są „niestrukturalne"). Tak więc reprezentatywność definicji jest warunkiem koniecznym opanowania konstrukcji specyfikacji wykazujących poprawność wyższego rzędu, w szczególności odpowiadających założeniom pragmatycznym.

Uzyskanie reprezentatywności definicji procesów wymaga zapewnienia:

- wykonalności operacji,
- wzajemnego wykluczenia przy korzystaniu z obiektów wspólnych (jedynymi takimi obiektami są zasoby wprowadzone w JS SDL),
- strukturalnej budowy grafu procesu.

Problem wykonalności operacji sprowadza się do problemów: wykonalności wejść oraz wykonalności pozostałych operacji. Ten drugi jest znany z programowania sekwencyjnego. Ciekawszy i specyficzny dla dekompozycji sieci na komunikujące się procesy jest natomiast problem wykonalności wejść. Jego rozwiązanie w postaci algorytmu konstrukcji tzw. sieci domkniętej (komunikacyjnie) przedstawiono w [1, 18] i zaimplementowano w opisanym dalej systemie SPRITE [13, 21, 10, 9]. Algorytm ten zapewnia, że wszystkie nadane sygnały zostaną odebrane we właściwych stanach — specyfikacja jest komunikacyjnie domknięta.

Przejście od specyfikacji do implementacji

Ostatecznym celem specyfikacji systemu jest jego implementacja zgodna z pierwotną specyfikacją. Zgodność oznacza, że zrealizowany system wykonuje zaplanowane funkcje i zachowuje się w sposób opisany przez specyfikację.

W związku z istniejącymi różnicami między językiem SDL a Chill (mimo bliskości modelu komunikacji procesów) nie jest możliwe podanie jednoznacznych, globalnych reguł automatycznej transformacji specyfikacji w SDL na programy w Chill. Można natomiast sformułować zbiór cząstkowych reguł i wskazówek opisujących sposób postępowania przy odwzorowywaniu wybranych konstrukcji języka SDL na konstrukcje w Chill. W tej sytuacji przejście od specyfikacji do implementacji, ze względu na znaczne pole dla arbitralnych decyzji, opiera się głównie na wiedzy, doświadczeniu i inwencji projektanta-programisty, które mogą być wspomagane systemem doradczym (por. np. [8]). Przejście od SDL do Chill jest naturalniejsze i prostsze niż w przypadku generowania na podstawie specyfikacji w SDL programu w języku C,C++ lub Pascal. Niewielka popularność języka Chill powoduje jednak, że dąży się do opanowania transformacji specyfikacji w SDL w programy w innych językach, a zwłaszcza w C++ [6], Ada i inne. Opracowuje się także wersje wykonalne SDL i kompilatory dla nich [6]. Te ostatnie prace mają na celu weryfikację specyfikacji i badanie charakterystyk ilościowych modeli systemów opisanych w SDL metodami symulacyjnymi (por. np. [26]).

Narzędzia

Język SDL może być w zasadzie stosowany bez wykorzystywania komputerowych narzędzi wspomagających. Dokumenty wchodzące w skład specyfikacji mogą być przygotowy-

wane ręcznie z użyciem szablonu do symboli graficznych. Jednak w zdecydowanej większości przypadków postępowanie takie nie jest praktyczne. Składa się na to kilka dość oczywistych przyczyn.

- Systemy o praktycznym znaczeniu są na tyle złożone, że objętość ich dokumentacji jest zbyt duża, by panować nad nią, opierając się tylko na procedurach ręcznych.
- Utrudnione jest wprowadzanie wszelkich zmian, manipulacja, przechowywanie i dystrybucja dokumentów.
- Zachowanie poprawności dokumentów (choćby tylko w sensie składniowym) staje się praktycznie niemożliwe.
- Bardzo trudne jest zachowanie spójności i integralności dokumentacji, z uwzględnieniem wzajemnych powiązań dokumentów składowych, zwłaszcza przy pracy zespołowej.

Stosowanie procedur ręcznych jest bardzo podatne na różnego rodzaju błędy wprowadzane we wczesnej fazie projektu, które mogą pozostawać niezauważone również w jego dalszych fazach, prowadząc do zwiększenia całkowitego kosztu projektu. W tej sytuacji stosowanie komputerowych narzędzi wspomagających staje się koniecznością.

Praktyczne stosowanie języka SDL, nawet do niewielkich projektów, wymaga więc oprogramowania wspomagającego, ułatwiającego konstruowanie, redagowanie i przechowywanie specyfikacji. Prowizorycznym rozwiązaniem jest adaptacja uniwersalnych pakietów graficznych (np. Autocad lub GEM Draw), przez zdefiniowanie odpowiedniej biblioteki symboli graficznych. Ułatwia to wytwarzanie, manipulowanie, przechowywanie i drukowanie diagramów, lecz nie zapewnia żadnej kontroli poprawności.

Profesjonalne posługiwanie się językiem SDL wymaga specjalnie wykonanego oprogramowania, które można kupić lub opracować samemu (nakład pracy rzędu 10-20 osobolat). W dalszej części artykułu opisano pakiet oprogramowania SPRITE opracowany w Instytucie Telekomunikacji Politechniki Warszawskiej.

Zakres funkcji środowiska wspomagającego specyfikację w SDL powinien odpowiadać typowym wymaganiom dla systemów CAD, z uwzględnieniem funkcji typowych dla narzędzi CASE (komputerowe wspomaganie wytwarzania oprogramowania). Oznacza to, że środowisko powinno mieć charakter graficzny i być wyposażone w funkcje:

- kontroli poprawności specyfikacji na różnych poziomach szczegółowości;
- zapewnienia spójności logicznej, aktualności i kompletności specyfikacji w czasie trwania projektu;
- zarządzania projektem (np. utrzymywanie wielu wersji);
- edycji graficznej, tekstowej i strukturalnej;
- wspomagania poprawnego projektowania (zarówno w sensie semantyki statycznej, jak i dynamicznej);
- wspomagania realizacji pewnej wybranej metodologii projektowania;
- wspomagania wytwarzania i archiwowania dokumentacji systemu;
- wspierania pracy zespołowej w sieci stanowisk roboczych;
- wspomagania przejścia od specyfikacji do implementacji.

Wszystkie wymienione funkcje powinny być zrealizowane i udostępniane w sposób zapewniający przyjazny kontakt z użytkownikiem.

Spełnienie — w odniesieniu do języka SDL — wszystkich przedstawionych wymagań (w szczególności tych związanych z zapewnieniem poprawności w sensie dynamicznym) jest zadaniem bardzo złożonym i trudnym do realizacji. Znane autorom pakiety oprogramowania realizują przytoczone postulaty w ograniczonym zakresie, uwzględniającym głównie funkcje edycji graficznej z prostą kontrolą poprawności składniowej oraz archiwowania, czyli do tych elementów, które są najprostrze do zaimplementowania.

Okazją do prezentacji oprogramowania wspomagającego są kolejne konferencje SDL Forum, odbywające się co dwa lata. Na ostatniej konferencji w 1989 roku w Lizbonie prezentowano 15 pakietów oprogramowania, z których tylko 6 jest dostępnych jako produkty komercyjne. Pozostałe narzędzia mają charakter eksperymentalny lub są przeznaczone tylko do użytku wewnętrznego. W sensie wymagań sprzętowych i programowych narzędzia takie są dostępne na komputerach klasy IBM PC i MacIntosh, Vax oraz graficznych stanowiskach roboczych Sun, Apollo, HP9000 w środowiskach programowych DOS, UNIX i Windows. Szczegółowe informacje na temat dostępnych narzędzi wspomagających można znaleźć w [6].

Środowiska wspomagające stosowanie języka SDL

SPRITE jest środowiskiem projektowym dla języka SDL (wg CCITT) i JS SDL, integrującym w jednym pakiecie oprogramowania zestaw funkcji, które wspomagają użytkownika przy:

- tworzeniu, redagowaniu, przechowywaniu i wyszukiwaniu poprawnych składniowo dokumentów w języku JS SDL, w postaci graficznej (GR) i tekstowej (PR);
- utrzymywaniu i kontroli wersji (konfigurowaniu) dokumentów projektu;
- translacji specyfikacji z postaci graficznej na tekstową: (SDL/GR→SDL/PR);
- wytwarzaniu dokumentacji;
- projektowaniu prawidłowego schematu komunikacji między procesami.

Stosowanie systemu SPRITE zwiększa wydajność projektanta przy specyfikowaniu i opisie oprogramowania dla central SPC, protokołów komunikacyjnych oraz innych systemów czasu rzeczywistego, przy minimalnych wymaganiach związanych ze szkoleniem użytkowników i ze sprzętem (sterowany przez menu „przyjazny" dla użytkownika tryb dialogu, wbudowane w program wyjaśnienia, standardowy komputer osobisty zgodny z IBM PC/XT lub AT). SPRITE wspomaga także użytkownika przy utrzymywaniu poprawności, integralności i spójności specyfikacji przez narzucenie dyscypliny projektowania.

Architektura systemu

Podstawowym pojęciem systemu SPRITE jest projekt. Obejmuje on specyfikacje wielu systemów, z uwzględnieniem wielu ich wersji (odmian lub faz zaawansowania projektu). Każdemu obiektowi strukturalnemu w JS SDL (sieć, system, blok, proces... itd.) odpowiada oddzielny dokument składający się z arkuszy. Arkusz jest obszarem o wielkości określonej przez format A4, przeznaczonym do umieszczania diagramów.

Dokumenty projektu są reprezentowane w specjalnej postaci nazywanej Uniwersalną Reprezentacją Pośrednią (URP), która umożliwia elastyczną współpracę funkcji SPRITE-a i zapewnia oszczędną formę ich przechowywania w pamięci zewnętrznej (typowy diagram — np. procesu — wypełniający arkusz zajmuje średnio około 2–3 KB; wielkość pliku reprezentującego arkusz w dużym stopniu zależy od długości atrybutów tekstowych symboli graficznych

Funkcje SPRITE-a są realizowane przez zbiór powiązanych ze sobą modułów oprogramowania, obejmujący: Konfigurator, Edytor Graficzny, Edytor Tekstowy, Analizator Składniowy, Translator SDL/GR → SDL/PR, Moduł Drukujący Dokumentację, Moduł Domknięcia Komunikacyjnego.

Konfigurator nadzoruje prawidłowość hierarchii obiektów strukturalnych JS SDL i umożliwia manipulowanie strukturą specyfikacji (zestawienie wersji, kopiowanie obiektów, odtwarzanie i zachowywanie fragmentów specyfikacji, operacje na plikach). Moduł ten spełnia także funkcję odwzorowywania hierarchii obiektów i ich arkuszy w odpowiednią strukturę organizacji plików na poziomie systemu operacyjnego MS-DOS.

Edytor Graficzny służy do redagowania diagramów w JS SDL/GR. Można jednocześnie redagować diagramy w kilku oknach (do czterech). Oferowany jest zestaw podstawowych funkcji: umieszczanie, usuwanie, przemieszczanie i kopiowanie pojedynczych symboli i grup symboli, łączenie symboli ręczne i automatyczne, zmiana wielkości symboli, wyszukiwanie symboli zawierających wskazany wzorzec tekstowy. Przeprowadzana jest kontrola składniowa łączonych symboli graficznych oraz elementów tekstowych z nimi związanych. Edytor ma wbudowany własny edytor tekstowy służący do redagowania elementów tekstowych symboli, będący uproszczoną wersją Edytora Tekstowego przeznaczonego do redagowania tekstów w JS SDL/PR, także z wbudowaną kontrolą składniową.

Edytor Tekstowy jest przeznaczony do redagowania dowolnych tekstów, w tym powstałych w wyniku translacji z postaci GR na PR. Ma on podstawowy zestaw funkcji właściwych dla typowego edytora ekranowego, obejmujący wyszukiwanie i zamianę wzorców oraz operacje na blokach. W edytor wbudowana jest funkcja analizy składniowej dla JS SDL/PR.

Translator przekształca wskazane fragmenty specyfikacji z postaci graficznej na tekstową. W wyniku translacji powstają pliki tekstowe, które mogą być poddane dalszej obróbce za pomocą Edytora Tekstowego.

Moduł Domknięcia Komunikacyjnego pozwala zaprojektować interakcję procesów w sposób gwarantujący, że wszystkie sygnały nadawane przez procesy zostaną odebrane z zachowaniem właściwej synchronizacji. Moduł jest luźno związany z systemem SPRITE w tym sensie, że operuje uproszczoną notacją SDL/PR przy opisie konstruowanego grafu komunikacji procesów.

Redagowanie graficzne

Każda specyfikacja należy do pewnego projektu, z którym jest związana biblioteka przechowująca definicje oddalone. Specyfikacja składa się z dokumentów, które dzielą się na arkusze. Podstawowym polem działania dla projektanta jest arkusz, przy czym jednocześnie może być widocznych kilka arkuszy, z których jeden w danej chwili jest przetwarzany.

Redagowanie odbywa się w trybie wielookienkowym. Arkusze dokumentu (do czterech dokumentów) są wyświetlane na ekranie w osobnych oknach graficznych. Okna są tworzone przez system w chwili otwarcia nowego arkusza, a usuwane po jego zamknięciu. Z edytorem graficznym jest zintegrowany ekranowy edytor tekstowy. Za każdym razem, gdy trzeba umieścić lub zmodyfikować na arkuszu elementy tekstowe, jest otwierane dodatkowe okno redagowania tekstu na tle okna graficznego.

Wszystkie elementy specyfikacji — graficzne i tekstowe — są po wprowadzeniu (dołączeniu) analizowane przez analizator syntaktyczny pod kątem poprawności składniowej. Umieszczanie właściwych symboli na arkuszu jest wymuszone przez zawartość menu dostosowaną do rodzaju specyfikowanej jednostki (sieć, system, blok, proces itd.). Przy łączeniu symboli — ręcznym lub automatycznym — badana jest zgodność powstających konstrukcji z regułami składni języka SDI. Analogicznie wszystkie wprowadzane elementy tekstowe są również poddawane (na żądanie) analizie składniowej.

Wszystkie dostępne funkcje edytora oraz dostępne na danym poziomie sybole są umieszczane w menu. Symbole graficzne można wybierać z graficznego menu przez naciśnięcie jednego klawisza. Poszczególne operacje edytora można uaktywniać po umieszczeniu kursora w odpowiednim miejscu arkusza, wybierając odpowiednią kombinację sterującą klawiszy (wyjaśnioną w menu).

Zbiór operacji edytora i zasady ich używania opracowano tak, by tworzenie i modyfikowanie specyfikacji było proste i szybkie. Sprzyja temu rastrowa organizacja ekranu. Na rysunku przedstawiono kopię ekranu podczas redagowania graficznego.

- **Operacje na oknach;**

— otwarcie i zamknięcie okna;
— wymiana arkusza w oknie (następny arkusz bieżącego dokumentu lub pierwszy arkusz nowego dokumentu), przejście w górę lub w dół hierarchii zagnieżdżenia jednostek specyfikacji;
— zmiana aktywnego okna.

- **Operacje na symbolach:**

— wstawianie i usuwanie symboli;
— wprowadzanie, wyświetlanie i poprawianie atrybutów tekstowych symboli;
— zwiększanie i zmniejszanie wielkości symboli;
— łączenie symboli (automatyczne lub ręczne);
— usuwanie połączeń między symbolami;
— redagowanie i analiza składniowa elementów tekstowych związanych z symbolem;
— markowanie i „odmarkowywanie" symboli;
— przenoszenie i kopiowanie symboli;
— zmiana atrybutów symbolu (np. zwrotu wyjść-wejść);
— definiowanie kanału.

- **Operacje pomocnicze:**

— drukowanie arkusza;
— wyświetlanie i ukrywanie rastra;
— zmiana skali;
— przemieszczanie kursora;
— ustalanie wersji;

- **Utrzymywanie i kontrola wersji**

W związku z planowanym opracowaniem systemu dokumentacji wyróżniono w środowisku narzędzie nazwane konfiguratorem. Do jego funkcji należy wybór wersji specyfikacji, która ma być poddana redagowaniu lub dla której ma być wytworzona dokumentacja, kopiowanie dokumentów. Wersja jest zdefiniowana przez właściwy (pod)zbiór arkuszy, który należy do niej. Konfigurator prezentuje także hierarchiczną strukturę specyfikacji i pozwala szybko poruszać się w hierarchii jednostek specyfikacji.

Transformacja z postaci graficznej na tekstową

Pomocniczą funkcją systemu jest translacja specyfikacji z postaci graficznej na postać tekstową. Dzięki istnieniu URP proces translacji znacznie się upraszcza, choć wymaga przyjęcia pewnych arbitralnych decyzji co do postaci tekstowej specyfikacji wynikowej. Funkcja ta jest potrzebna przy przenoszeniu specyfikacji na systemy komputerowe, które mają inne (lub w ogóle nie mają) możliwości pracy w trybie graficznym i może być pomocna przy tworzeniu dokumentacji.

Domknięcie komunikacyjne

W trakcie konstrukcji specyfikacji komunikujących się procesów pojawia się następujący problem: po wyspecyfikowaniu w procesie *A* wyjścia wysyłającego sygnał do procesu *B* należy określić, gdzie trzeba i wystarczy wyspecyfikować wejścia dla tego sygnału, by każde wejście było wykonalne i by w każdym stanie, w którym sygnał ten może być odebrany, było wyspecyfikowane odpowiednie wejście.

Algorytm — wskazujący wszystkie takie miejsca — jest oparty na twierdzeniu o domknięciu i dwóch regułach projekcji, sformułowanych i udowodnionych w rozprawie doktorskiej P. Ostrowskiego i opisanych także w [3, 1, 15]. Algorytm ten jest zaimplementowany w systemie SPRITE, w sposób opisany w [1]. Moduł realizuje funkcję automatycznego egzekwowania algorytmu domknięcia — po wyspecyfikowaniu wyjścia, konieczne wejścia są specyfikowane automatycznie.

Okazuje się, że zapewnienie poprawnej struktury sterowania komunikacją między procesami o dużym stopniu niezależności jest często zadaniem trudnym i uciążliwym, nawet przy dysponowaniu metodami formalnymi.

Zaimplementowana wersja języka

Dla potrzeb własnych i Jednolitego Systemu Środków Technicznych Komutacji opracowano w Instytucie Telekomunikacji Politechniki Warszawskiej (na zlecenie TELKOM-ZWUT) język specyfikacji, nazwany skrótowo JS SDL [15, 6]. Język ten stanowi rozszerzenie podzbioru języka SDL określonego zaleceniem CCITT Z.100. Wyznaczając przyjęty podzbiór SDL miano na uwadze poprawienie opanowywalności konstrukcji specyfikacji w procesie projektowania systemu telekomunikacyjnego, na poziomach abstrakcji bliższych implementacji, a więc przeznaczonych dla projektantów urządzeń.

W tym celu:

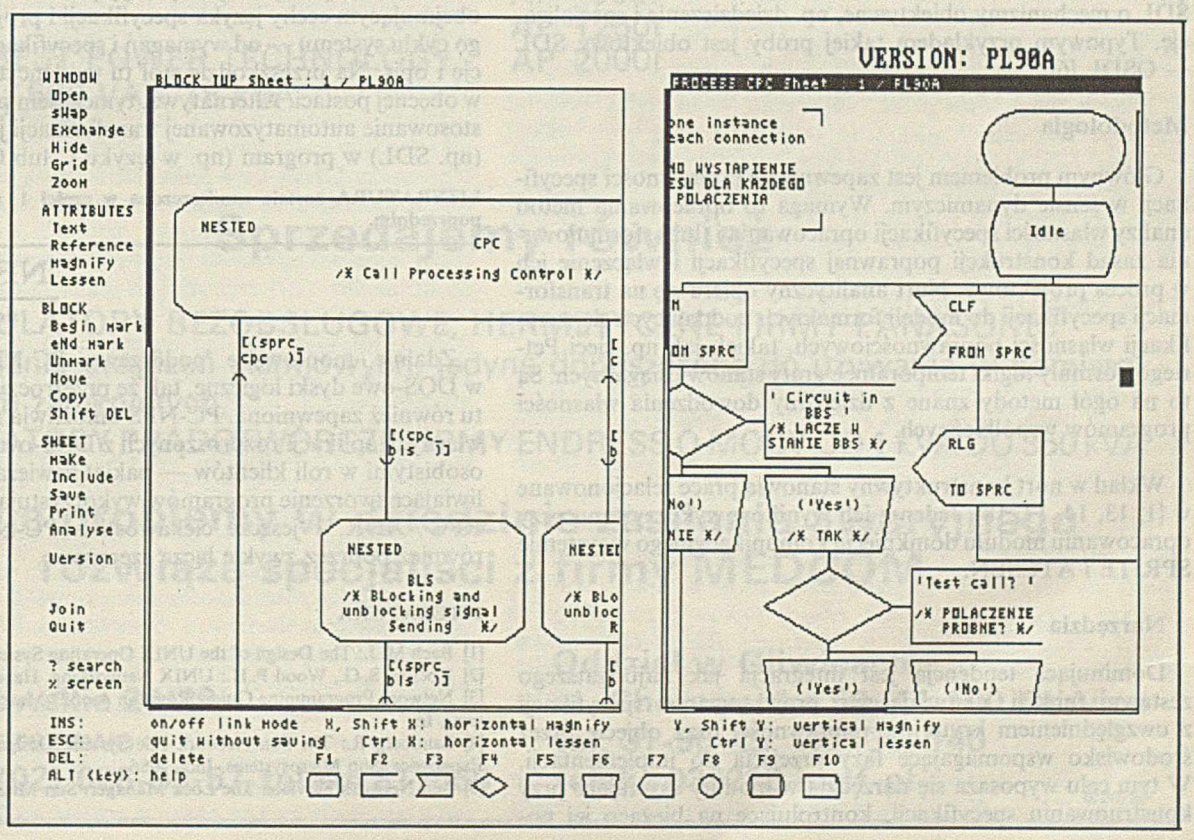
— usunięto konstrukcje niebezpieczne w stosowaniu — łatwo prowadzące do błędów i utraty panowania nad projektem (zwłaszcza wielowariantowe środki komunikacji między procesami), zachowując jedynie komunikację przez sygnały;
— zrezygnowano z wielowariantowości środków językowych, usuwając konstrukcje nadmiarowe — warunek umożliwiający dekompozycję procesu na usługi;
— uproszczono nieco sposób definiowania typów danych i ujednolicono składnię wyrażeń.

Przyjęte rozszerzenia tak uzyskanego podzbioru SDL, mają ten sam cel ogólny i polegają na:

- wprowadzeniu pojęcia zasobów i puli zasobów;
- wyróżnieniu konwencji interpretacyjnych (sieci jednorodnej i niejednorodnej) [25, 6, 15];
- wprowadzeniu możliwości adresowania bezpośredniego sygnałów;
- rozszerzeniu stosowalności języka także do specyfikacji otoczenia systemu (wprowadzenie definicji sieci — universe — złożonej z definicji systemu i otoczenia, wprowadzenie wejść inicjatywnych);
- wprowadzeniu uproszczonej metody specyfikowania time-out'ów obejmujących jeden stan i opóźnień w przejściu.

Oprócz powyższych ograniczeń i rozszerzeń wprowadzono drobniejsze usprawnienia wzbogacające możliwości organizowania dokumentacji i wizualizacji specyfikacji za pomocą komputera.

W ostatnio opracowanej, rozszerzonej komercyjnej wersji systemu, zwanej ATCSEN, zaimplementowano także wersję składniową SDL zgodną z zaleceniem CCITT Z.100 [4].



Przykładowy ekran systemu SPRITE podczas edycji graficznej

Tendencje rozwojowe

Prezentacja tendencji rozwojowych wymaga pewnej wstępnej roboczej klasyfikacji. Obejmuje ona następujące zagadnienia:
— język SDL,
— metodologię jego stosowania,
— narzędzia.
Wszystkie wymienione elementy są ze sobą powiązane i wzajemnie zależne.

Język

W tym przypadku mamy do czynienia ze sprzecznymi tendencjami. Z jednej strony CCITT, który jest organizacją standaryzacyjną, dąży do ograniczenia jakichkolwiek zmian i stabilizacji zalecenia Z.100, z drugiej natomiast doświadczenia użytkowników i implementatorów języka oraz postępy w dziedzinie inżynierii oprogramowania skłaniają do wprowadzenia zmian „uszlachetniających" SDL.

W wyniku studiów przeprowadzonych w IT PW opracowano wersję języka SDL stanowiącą jego podzbiór uzupełniony o pewne dodatkowe pojęcia i mechanizmy, takie jak np. sieć jednorodna, zasoby i związane z nimi operacje podstawowe, wejście inicjatywne, jawna definicja otoczenia [18]. Dodatkowo opracowano także wersję przeznaczoną do badań symulacyjnych systemów na podstawie ich specyfikacji (język SDLSIM [26]).

Jednym z przedmiotów badań jest próba rozbudowy języka SDL w sposób umożliwiający również jego stosowanie w fazie implementacji, czyli uczynienie z niego tzw. języka szerokopasmowego (ang. *wide-spectrum*). Wymaga to uzupełnienia o środki typowe dla języka programowania. Decyzja o tym, jakie są to środki, jest uwarunkowana przez wybrany styl programowania — proceduralny, obiektowy lub deklaratywny. Podjęto próby rozszerzeń we wszystkich wymienionych kierunkach [25, 19].

Popularnym nurtem rozwojowym jest rozszerzenie języka SDL o mechanizmy obiektywne, np. dziedziczenie i specjalizację. Typowym przykładem takiej próby jest obiektowy SDL — OSDL [6].

Metodologia

Głównym problemem jest zapewnienie poprawności specyfikacji w sensie dynamiczym. Wymaga to opracowania metod analizy własności specyfikacji opracowania i (lub) sformułowania zasad konstrukcji poprawnej specyfikacji i włączenie ich w proces projektowy. Nurt analityczny opiera się na transformacji specyfikacji do modeli formalnych poddających się weryfikacji własności poprawnościowych, takich jak np. sieci Petriego, formuły logiki temporalnej, grafy stanów osiągalnych. Są to na ogół metody znane z dziedziny dowodzenia własności programów współbieżnych.

Wkład w nurt konstruktywny stanowią prace relacjonowane w [1, 13, 14, 17, 18]. Jeden z ich wyników wykorzystano przy opracowaniu modułu domknięcia komunikacyjnego w systemie SPRITE i ATCSEN.

Narzędzia

Dominującą tendencją jest integracja jak najbogatszego zestawu funkcji ułatwiającego projektowanie specyfikacji z uwzględnieniem kryteriów poprawności oraz objęcie przez środowisko wspomagające fazy przejścia do implementacji. W tym celu wyposaża się narzędzia w moduły asystujące przy konstruowaniu specyfikacji, kontrolujące na bieżąco jej poprawność, w moduł symulatora lub interpretacji specyfikacji oraz generowania wykonywalnego kodu w popularnych językach programowania, np. C, C++, Pascal, Chill.

W IT PW podjęto prace nad wykonywalną specyfikacją i jej dołączeniem do zestawu narzędzi systemu SPRITE [22].

Problemem, który także wymaga rozwiązania, jest zapewnienie przenośności dokumentów w SDL/GR między różnymi systemami. Na razie możliwość taka istnieje tylko na poziomie SDL/PR.

* * *

Wymagania środowiska telekomunikacyjnego prowadzą do konieczności stosowania formalnych środków specyfikacji i opisu funkcjonalnego systemów, oraz stosowania tych środków w procesie projektowania.

Język SDL, oparty na koncepcji sieci komunikujących się procesów, rozumianych jako rozszerzone automaty skończone (EFSM), komunikujące się za pomocą sygnałów przez nieograniczone kolejki FIFO, z dobrze zdefiniowaną składnią i semantyką i — co szczególnie istotne w praktyce — z równoważną, sugestywną składnią graficzną, zdobył najszerszy zakres zastosowań i popularność. Inne języki, jak np. Estelle [20] lub Lotos, nie mają wersji graficznej i są stosowane w wąskich kręgach użytkowników.

Z uwagi na uniwersalność modelu, sugestywną notację graficzną, możliwość wyrażania uwarunkowań czasowych oraz dostępność oprogramowania wspomagającego, SDL może znaleźć wiele zastosowań pozatelekomunikacyjnych, zwłaszcza w dziedzinie specyfikacji i projektowania systemów czasu rzeczywistego zagnieżdżonych (ang. *embedded*) w otoczeniu.

Stosowanie SDL (lub języków pochodnych, np. JS SDL) w projektowaniu systemów wymaga dalszego rozwoju bazy teoretycznej, metodologicznej i narzędziowej, mimo jej dość znacznego zaawansowania. Prawdopodobny rozwój będzie zmierzał do pokrycia jednym językiem szerokopasmowym, obejmującym cechy języka specyfikacji i programowania, całego cyklu systemu — od wymagań i specyfikacji po implementację i opis. Na przeszkodzie stoi tu tendencja zamrożenia SDL w obecnej postaci. Alternatywą, tymczasem popularniejszą, jest stosowanie automatyzowanej transformacji języka specyfikacji (np. SDL) w program (np. w języku C lub C++).

LITERATURA zostala zamieszczona w części 1. artykułu, w numerze poprzednim.

NFS od środka

dokończenie ze s. 38

Zdalnie montowane poddrzewa PC-NFS odwzorowuje w DOS-owe dyski logiczne, tak że przezroczystość dostępu jest tu również zapewniona. PC-NFS umożliwia również tworzenie własnych aplikacji rozproszonych z DOS-owymi komputerami osobistymi w roli klientów — pakiet zawiera narzędzia umożliwiające tworzenie programów wykorzystujących mechanizmy RPC / XDR. I jeszcze ciekawostka: PC-NFS umie działać również poprzez zwykłe łącza szeregowe.

LITERATURA

[1] Bach M. J.: The Design of the UNIX Operating System. Prentice-Hall, 1986
[2] Kochan S. G., Wood P. H.: UNIX Networking. Hayden Books, 1989
[3] Network Programming Guide. Revision A of 27 March, 1990. Sun Microsystems, In.
[4] Sandberg R.: The Sun Network File System: Design, Implementation and Experience. Sun Microsystems, Inc., 1986
[5] Sun Network Services: The Lock Manager. Sun Microsystems, Inc., 1987

JACEK SURMA

MEDCOM

CHCESZ ZABEZPIECZYĆ BAZY DANYCH PRZED UTRATĄ INFORMACJI?

CHCESZ ZAPEWNIĆ PRAWIDŁOWĄ PRACĘ

KOMPUTERA * CENTRALI TELEFONICZNEJ * TELEFAKSÓW

W WARUNKACH ZANIKU NAPIĘCIA ZASILAJĄCEGO

kup ZASILACZE BEZPRZERWOWE

naszej produkcji
UPS – 140 (abonenckie centrale telefoniczne i telefaksy)
UPS – 450
UPS – 850

produkcji BEST POWER TECHNOLOGY
ON – LINE: 500 VA – 18 kVA

produkcji
AMERICAN POWER CONVERSION:

AP 400I
AP 600I
AP 900I
AP 1250I
AP 2000I
AP 800I
AP 1200I

Sprzedajemy również

* AKUMULATORY BEZOBSŁUGOWE, HERMETYCZNE FIRMY PANASONIC do zasilania urządzeń alarmowych, jedyne dopuszczone do używania w pomieszczeniach zamkniętych,
* GENERATORY PRĄDOTWÓRCZE FIRMY ENDRESS O MOCY OD 1 kVA DO 350 kVA

Twoje problemy w dziedzinie zasilania awaryjnego rozwiążą specjaliści z firmy MEDCOM.

BIURO
Al. Ujazdowskie 26/39
00-478 Warszawa
tel., faks (02) 628-93-57, teleks 817060

Oddział w Gliwicach
ul. Chorzowska 50, 44-100 Gliwice
tel. 31-90-45 w. 139, 140
teleks 036215 iptif pl

0/20/90

/jak unix z unix-em

Studium klienta i usługodawcy, czyli o Suno-owej koncepcji sieci otwartych (3)

NFS od środka

Network File System (NFS), kluczowy element Sun-owej koncepcji sieci otwartych (por. INFORMATYKA nr 8, 1991), jest rozbudowaną usługą sieciową, umożliwiającą użytkownikom wygodne i sprawne korzystanie ze zdalnych plików. Dostęp do zdalnego pliku staje się możliwy po dowiązaniu zdalnego poddrzewa, do którego plik należy, do lokalnej hierarchii katalogów. Od tej pory na pliku tym można działać niemal dokładnie tak, jak na lokalnym, bez konieczności przesyłania go w całości. Wśród innych systemów tego typu NFS wyróżnia się przenośnością i przystosowaniem do pracy w sieciach heterogenicznych.

Projektantom NFS-u przyświecało kilka trudnych do pogodzenia celów [4]. Po pierwsze, zakładało brak bliższych związków z jakimkolwiek konkretnym systemem operacyjnym czy rodziną komputerów. Protokół miał być na tyle prosty i ogólny, by dawał się implementować zarówno w VAX-ach pod VMS-em, jak i na komputerach osobistych pod DOS-em. Po drugie, należało zapewnić łatwość podnoszenia się po awariach, zarówno po stronie klientów, jak i usługodawców. Trzecim celem była przezroczystość: aplikacje nie powinny zauważać różnic w dostępie do plików lolalnych i zdalnych. Po czwarte, należało zapewnić możliwość utrzymania pełnej UNIX-owej semantyki zdalnych plików. Ostatnim celem była szybkość dostępu, co najmniej taka, która by zniechęcała użytkowników do sprowadzania zdalnych plików w całości.

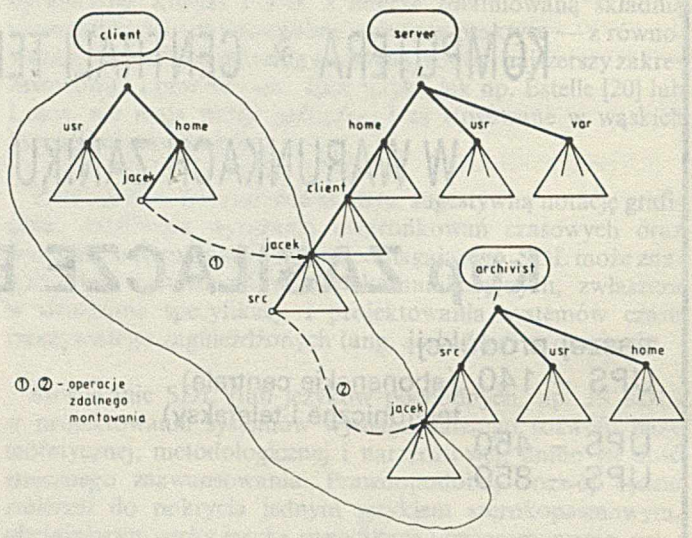
Projektanci NFS-u przekroczyli plan — nie tylko powstał produkt spełniający większość oczekiwań, lecz przy okazji również UNIX wzbogacił się o nowy, niezwykle użyteczny mechanizm: **wirtualny system plików (VFS)**. Zanim zajmiemy się bliżej wnętrzem i mechaniką NFS-u (i VFS-u), spójrzmy najpierw na NFS od trochę innej strony.

OCZAMI UŻYTKOWNIKA

Wyobraźmy sobie sieć lokalną, a w niej trzy komputery o nazwach *client*, *server* i *archivist*. Pierwszy z nich to stacja robocza, pozostałe do duże komputery z potężnymi dyskami. Załóżmy, że wszystkie komputery pracują pod UNIX-em i dysponują NFS-em.

Przyjmijmy, że hierarchia katalogów każdego z systemów jest taka, jak na rys. 1. Przyjmijmy, że przykładowy użytkownik **jacek** na dysku swej lokalnej stacji *client* ma pusty katalog osobisty **/home/jacek**, część swoich plików przechowuje na dysku komputera *server* w katalogu **/home/client/jacek**, a część na dysku komputera *archivist* w katalogu **/src/jacek**. I chciałby mieć wygodny dostęp do wszystkich swoich plików. Co powinien zrobić?

Po pierwsze upewnić się, że komputery *server* i *archivist* są skłonne udostępnić (wyeksportować) interesujące go katalogi komputerowi *client*. Lista katalogów przeznaczonych do udostępnienia przez dany komputer mieści się w pliku **/etc/exports**, który w razie potrzeby należy poddać edycji. Załóżmy, że katalogi są na odpowiednich listach.



Rys. 1. Przykład NFS-owej hierarchii katalogów

Następnym krokiem jest dowiązanie zdalnych poddrzew do lokalnej hierarchii katalogów naszego użytkownika. Służy do tego znane UNIX-owe polecenie **mount**, rozszerzone o możliwość montowania zdalnych poddrzew katalogów:

client! *mount -t nfs server:/home/client/jacek /home/jacek*
client! *mount -t nfs archivist:/src/jacek /home/jacek/src*

Zauważmy, że drugi punkt montowania, /home/jacek/src, sam jest zdalnym katalogiem, dostępnym dzięki pierwszej operacji montowania.

Od tej pory wszystkie pliki naszego użytkownika należą do jednej logicznej hierarchii katalogów. Użytkownik ma normalny dostęp do zdalnych plików, taki sam jak do lokalnych.

Mniej doświadczony użytkownik w ogóle nie musi nic wiedzieć ani o eksportowaniu, ani o montowaniu[1]. Administrator systemu dysponuje wystarczającymi środkami (plik **/etc/fstab**, bazy danych **automontera**), by zapewnić użytkownikowi permanentny dostęp do wszystkich potrzebnych mu plików, bez względu na to, gdzie się one znajdują.

[1] W przyzwoicie administrowanym systemie prawo do wykonania polecenia *mount* oraz do edycji plików z /etc i tak przysługuje tylko superużytkownikowi.

Między klientem i usługodawcą

Jak widać, przeciętny użytkownik stacji roboczej wcale nie musi być świadom tego, że korzysta z NFS. Inaczej ma się rzecz z administratorami sieci oraz z programistami przystosowującymi swoje programy do pracy w środowisku NFS-owskim. Ci pierwsi muszą rozumieć mechanizm działania NFS-u i rolę tych jego elementów, które stanowią przedmiot administrowania (pliki systemowe, demony), oraz mieć świadomość NFS-owych ograniczeń. Drudzy powinni co najmniej zdawać sobie sprawę z subtelnych różnic w operowaniu plikami zdalnymi i lokalnymi. Zarówno jednym, jak i drugim przyda się więc pewna wiedza o NFS-owym wnętrzu i mechanice działania.

Ujmując rzecz najprościej, konstrukcja NFS wspiera się na trzech elementach — **procedurach dostępu** po stronie klienta, **procesie usługodawcy** i **protokole**. Piersze dwa z nich mogą różnić się istotnie w różnych implementacjach, trzeci — protokół rozmowy klienta z usługodawcą — pozostaje niezmienny. Zacznijmy więc od protokołu.

Protokół

Bez mechanizmu zdalnych procedur nie byłoby NFS-u. Zdecydowano o tym już właśnie na poziomie protokołu, który zdefiniowano w **języku RPC/XDR** (por. INFORMATYKA nr 8, 1991), a więc w kategoriach zdalnych procedur, ich argumentów, wyników i efektów. Synchronizm zdalnych procedur w RPC/EDR wielce ułatwia zarówno implementację, jak i samo zrozumienie protokołu.

NFS-owy protokół jest **bezstanowy**. Znaczy to tyle, że argumenty każdej z procedur niosą komplet danych potrzebnych do jej zdalnego wykonania, a po stronie usługodawcy nie przechowuje się żadnej informacji dotyczącej stanu klientów czy historii wywołań. Bezstanowość protokołu, a więc w powyższym sensie również usługodawcy, który jest bezpośrednią przyczyną niemal wszystkich NFS-owych ograniczeń — o czym za chwilę — okazuje się nieocenionym dobrodziejstwem w przypadku **awarii**. Gdy awarii ulegnie komputer usługodawcy, strona klienta może po prostu w utalonych odstępach czasu [2] ponawiać swe żądanie, na przykład aż do chwili, gdy usługodawca znów podejmie działanie. Natomiast awaria komputera klienta nie wymaga w ogóle żadnych akcji po stronie usługodawcy.

W przypadku protokołu stanowego realizacja prawidłowej reakcji na awarie jest nieporównywalnie trudniejsza. Usunięcie awarii komputera usługodawcy oznacza dla klienta konieczność odbudowania tam informacji o swoim stanie, z kolei awaria komputera klienta zmusza usługodawcę do usunięcia tejże informacji.

Oczywiście bezstanowość NFS-owego protokołu nie likwiduje wszelkich poawaryjnych kłopotów. Operacje na plikach są z natury stanowe i nie można ich bezkarnie przerywać ani powtarzać. Natomiast chodzi o to, by sam protokół nie komplikował znakomicie i tak trudniejszego problemu. Bezstanowość protokołu ma zresztą jeszcze jedną ważną zaletę — umożliwia korzystanie w warstwie transportowej z protokołu UDP, znacznie efektywniejszego niż TCP. Ponieważ datagramowy **UDP** może gubić pakiety, strona klienta bywa zmuszona wysyłać je ponownie, co byłoby nie do przyjęcia, gdyby usługodawca działał stanowo.

Częstym argumentem procedu NFS-owego protokołu jest **uchwyt pliku** (ang. *file handle*), struktura służąca do identyfikacji zdalnego pliku, przekazywana klientowi przez usługodawcę. Klient nie interpretuje zawartości uchwytu, ma ona znaczenie wyłącznie dla usługodawcy.

Pora przedstawić sam protokół, czyli kilkanaście składających się nań procedur — oczywiście w uproszczeniu, pełną definicję w języku RPC/XDR można znaleźć w [3]. Na poniższej liście *up* oznacza uchwyt, *ukat* — uchwyt katalogu, *plik* i *katalog* to nazwy pliku lub katalogu, a określenie plik *up* i katalog *ukat* oznacza plik i katalog *identyfikowany przez up* lub *przez ukat*.

null( ) → ( ) — procedura pusta, służy do mierzenia czasu oczekiwania na odpowiedź,
getattr(up) → (atrybuty) — zwraca atrybuty pliku *up*,
setattr(up, atrybuty) → (atrybuty) — ustawia tryb, *uid*, *gid*, długość oraz czasy dostępu i modyfikacji pliku *up*,
lookup(ukat, plik) → (up, atrybuty) — zwraca uchwyt i atrybuty *pliku* w katalogu *ukat*,
readlink(up) → (napis) — zwraca napis zawarty w pliku powiązania symbolicznego *up*,
read(up, miejsce, n) → (atrybuty, dane) — zwraca atrybuty oraz *n* znaków począwszy od danego miejsca w pliku *up*,
write(up, miejsce, n, dane) → (atrybuty) — zapisuje do pliku *up n* znaków począwszy od danego *miejsca* i zwraca atrybuty,
create(ukat, plik, atrybuty) → (up, atrybuty) — tworzy w katalogu *ukat* nowy *plik* i zwraca jego uchwyt i atrybuty,
remove(ukat, plik) → (status) — usuwa *plik* z katalogu *ukat*,
rename(ukat, plik, UKAT, PLIK) → (status) — przemieszcza *plik* z katalogu *ukat* do katalogu *UKAT* i zmienia nazwę *pliku* na *PLIK*,
link(ukat, plik, UKAT, PLIK) → (status) — tworzy w katalogu *UKAT* nowy *PLIK* stanowiący powiązanie z *plikiem* z katalogu *ukat*,
symlink(ukat, plik, napis) → (status) — tworzy w katalogu *ukat* nowy *plik* powiązania symbolicznego i wiąże z nim *napis* (usługodawca nie interpretuje napisu),
mkdir(ukat, katalog, atrybuty) → (up, atrybuty) — tworzy nowy katalog w katalogu *ukat* i zwraca uchwyt nowego katalogu oraz jego atrybuty,
rmdir(ukat, katalog) → (status) — usuwa pusty katalog z katalogu *ukat*,
readdir(ukat, kolejna, n) → (pozycja) — zwraca *n* znaków kolejnej pozycji katalogu *ukat*; każda pozycja zawiera nazwę pliku, jego identyfikator, oraz nieinterpretowany wskaźnik do następnej pozycji danego katalogu; zerowa *kolejna* pozycja oznacza odczyt począwszy od pierwszej pozycji w katalogu,
statfs(up) → (info) — zwraca informację (wielkości bloku, liczba wszystkich bloków, liczba dostępnych, itp.) o systemie plików, w którym rezyduje plik *up*.

Z analizy protokołu wynikają dwa spostrzeżenia. Pierwsze dotyczy NFS-owego **modelu zdalnego systemu plików**, czyli tego, jak NFS wyobraża sobie system plików, z którym ma do

[2] Notabene te interwały muszą być wystarczająco długie, by nie dochodziło do nieuzasadnionego ponawiania żądań (groźne zwłaszcza w przypadku zapisu!), na przykład, gdy jedynym powodem braku odpowiedzi jest nadmierne obciążenie komputera usługodawcy.

czynienia NFS-owy usługodawca. Widać, że nie zakłada się zbyt wiele:

- system plików jest **hierarchiczny**, tworzy drzewo katalogów z plikami jako liśćmi i z pojedynczym korzeniem,
- pliki i katalogi mają **nazwy**, które są w ramach każdego z katalogów jednoznaczne,
- nie zakłada się **żadnej wewnętrznej struktury** plików, katalogi wiążą nazwy plików z ich **numerami**, jednoznacznymi w ramach danego systemu plików (uchwyt identyfikuje plik dla klienta, numer dla usługodawcy),
- każdy tak rozumiany system plików rezyduje **w całości na jednym komputerze.**

Taki model ma niewątpliwie UNIX-owy rodowód, ale jest na tyle ogólny, że zawiera w sobie nawet najbardziej egzotyczne systemy plików, o ile tylko operują one pojęciami pliku i katalogu. Oczywiście różne systemy operacyjne mogą nakładać rozmaite ograniczenia na przykład na długość nazwy pliku czy głębokość drzewa katalogów.

Drugie spostrzeżenie jest dosyć zaskakujące: NFS-owy **protokół nigdy nie operuje ścieżkami**, a wyłącznie nazwami w ramach określonego katalogu. Powody są dwa: ideologiczny i techniczny. Pierwszy sprowadza się do tego, że operowanie ścieżkami już na poziomie protokołu mogłoby prowadzić do wbudowania weń pewnych założeń co do ich postaci, a to ograniczyłoby jego uniwersalność. Zauważmy przy tym, że operowanie ścieżkami jest nieuniknione **tylko podczas montowania**. Z tego względu zdecydowano się stworzyć dla operacji montowania **odrębny protokół**, który mógłby być rozmaicie implementowany lub nawet zmieniany — bez wpływu na działanie samego NFS-u. Natomiast drugi powód — który wynika m. in. z założenia bezstanowości usługodawcy — powinien stać się zrozumiały już niebawem.

Po stronie usługodawcy

Zajmijmy się teraz NFS-owym usługodawcą, który w klasycznej UNIX-owej implementacji jest demonem, nazywa się najczęściej **nfsd** i występuje zwykle w gromadzie. Jego rola sprowadza się do oczekiwania na żądania klientów, realizacji żądań i odsyłania wyników, przy czym owe żądania to nic innego, jak zdalne wywołania procedur z NFS-owego protokołu. Ponieważ NFS-owi usługodawcy działają sekwencyjnie, więc dla zwiększenia przepustowości uruchamia się ich od razu kilka. Każdy z nich startuje jako proces użytkowy, wywołuje funkcję systemową nfsd () i nigdy nie wraca — kontekst użytkowy jest tu potrzebny wyłącznie po to, by **nfsd** mógł spać oczekując na żądania[3].

Wiemy już, że do identyfikacji plików udostępnianych klientom usługodawca używa struktur zwanych uchwytami, gdyż numery i-węzłów — jednoznaczne tylko w ramach danego systemu plików — w sieci już by nie wystarczyły. Uchwyt to trójka liczb: numer i-węzła, **numer generacji i-węzła** i **identyfikator systemu plików**. Wprowadzenie tych dwóch ostatnich wymagało rozszerzeń w UNIX-owych strukturach systemowych — odpowiednio w i-węźle i w superbloku. Numer generacji jest niezbędny w sytuacji, gdy usługodawca przekazuje klientowi uchwyt pliku, który to plik zaraz potem usuwa inny klient. I-węzeł po zwolnieniu mógłby zostać następnie związany z zupełnie innym plikiem, niż ten, który interesował pierwszego z klientów. Zwiększenie numeru generacji przy każdym zwalnianiu odpowiadającego mu i-węzła likwiduje ten problem[4]. Z kolei wspomniany identyfikator pozwala usługodawcy wybrać ten z pozostających w jego gestii systemów plików, który interesuje klienta.

NFS rozszerza na pliki zdalne klasyczny UNIX-owy styl **kontroli uprawnień**. Mechanizm RPC/XDR umożliwia przekazywanie usługodawcy parametrów identyfikacyjnych klienta — *uid* i *gid* — wraz z każdym wywołaniem zdalnej procedury. Warunkiem skuteczności takiego rozwiązania jest zachowanie w całej sieci jednolitej przestrzeni (*uid, gid*), co w przypadku większych sieci stanowiłoby zadanie niewątpliwie przekraczające cierpliwość przeciętnego administratora — gdyby nie inna Sun-owa usługa sieciowa: **Network Information Service** (NIS), znana dawniej jako **Yellow Pages** (YP). Ten ważny i ciekawy pakiet zasługuje na osobne omówienie, być może w następnym odcinku niniejszego cyklu.

Po stronie klienta

Sun-Owa implementacja strony klienta udostępnia użytkownikom NFS-u niemal idealnie **przezroczysty**[5] dostęp do zdalnych plików. Kluczowym elementem tej przezroczystości jest zachowanie tradycyjnego UNIX-owego **formatu ścieżek** również w przypadku plików zdalnych. To właśnie z tego powodu zdecydowano się dokonać rozszerzenia starej UNIX-owej operacji montowania tak, by można było montować zdalne poddrzewa. Wcześniejsze sieciowe systemy plików specyfikowały pliki zdalne przy pomocy ścieżek typu *komputer/ścieżka* lub */../ komputer / ścieżka*, co oczywiście nie miało nic wspólnego z przezroczystością. Jedyną wadą rozwiązania Sun-owego jest to, że zdalny plik staje się dostępny dopiero po zamontowaniu odpowiedniego poddrzewa. Wadę tę łagodzi zresztą korzystanie z **automontera** (ang. *automounter*), ściśle powiązanej z NFS-em usługi sieciowej. Automonter bierze na siebie odpowiedzialność za to, by poddrzewo ze zdalnymi plikami, których zamierzamy używać, zostało uprzednio zamontowane. Wiedzę o tym, co i gdzie zamontować, czerpie, ze swoich baz danych, które przygotowuje administrator systemu.

Aby zrozumieć, jak strony klienta i usługodawcy współpracują w realizacji owej przezroczystości, należy przyjrzeć się bliżej zainspirowanej przez NFS UNIX-owej nowince: wirtualnemu systemowi plików.

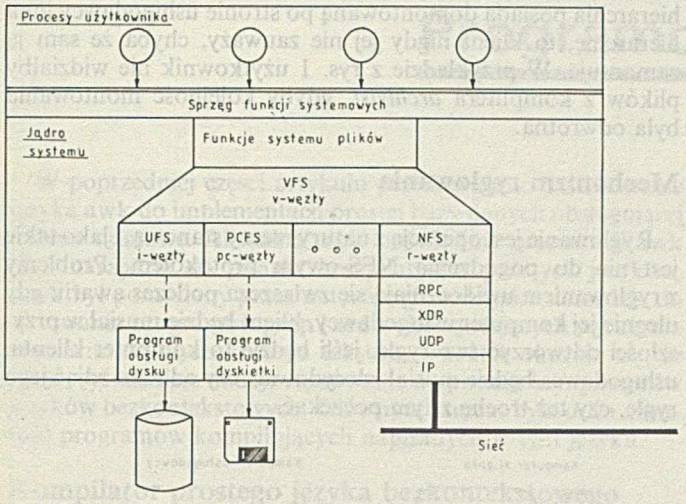
Wirtualny system plików

VFS [2, 3, 4] najlepiej wyobrażać sobie jako **nową warstwę** w jądrze UNIX-a, wstawioną gdzieś pomiędzy funkcje systemowe odnoszące się do plików a i-węzły (rys. 2). Do tej pory UNIX miał tylko jeden typ systemu plików (w dwóch odmianach: klasycznej i BSD), VFS umożliwia dołączenie nowych typów trochę tak, jak dołącza się nowe programy obsługi urządzeń.

[3] Istnieje również **UNFS** — „wzorcowa" implementacja NFS-owego usługodawcy w postaci procesu użytkownika. Właśnie UNFS jest zwykle punktem wyjścia do przenosin NFS-u pod inne systemy operacyjne [4].

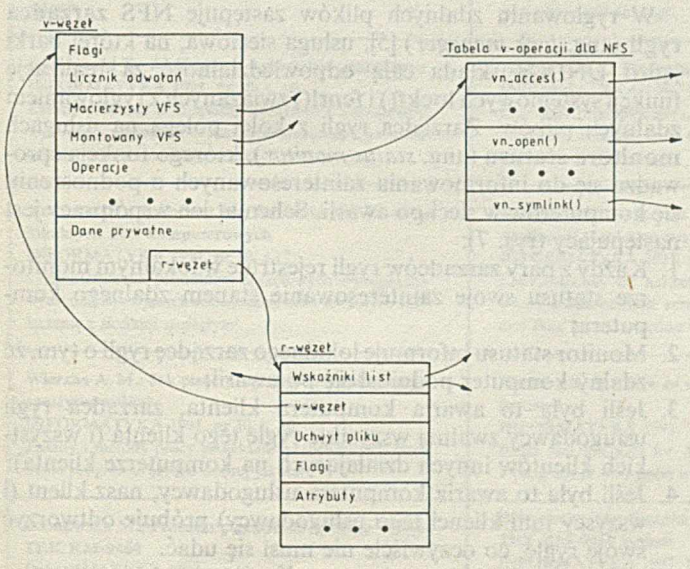
[4] Zbliżony mechanizm znajdziemy np. również w UNIX-owej implementacji semaforów, skrzynek pocztowych i regionów dzielonych [1].

[5] Całkowita przezroczystość w przypadku bezstanowego protokołu nie jest możliwa. Patrz niżej.



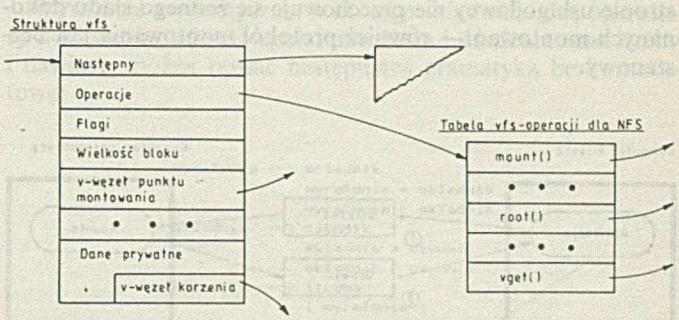
Rys. 2. Wirtualizacja systemu plików

Implementacja VFS-u wprowadza do jądra UNIX-a dwa nowe typy struktur: węzły wirtualne, czyli **v-węzły** i **vfs-struktury**. Głównym powodem ich wprowadzenia była potrzeba ukrycia przed funkcjami systemowymi tego, że w rzeczywistości mają do czynienia z systemami plików różnych typów. Funkcje systemowe operują na v-węzłach i vfs-strukturach w jednolity sposób, a do wszelkich operacji specyficznych dla danego typu systemu plików wywołują nieznane sobie funkcje wskazywane pośrednio przez odpowiednie pola w v-węzłach lub vfs-strukturach. (Inaczej mówiąc v-węzły i vfs-struktury implementują coś w rodzaju abstrakcyjnych struktur danych.)



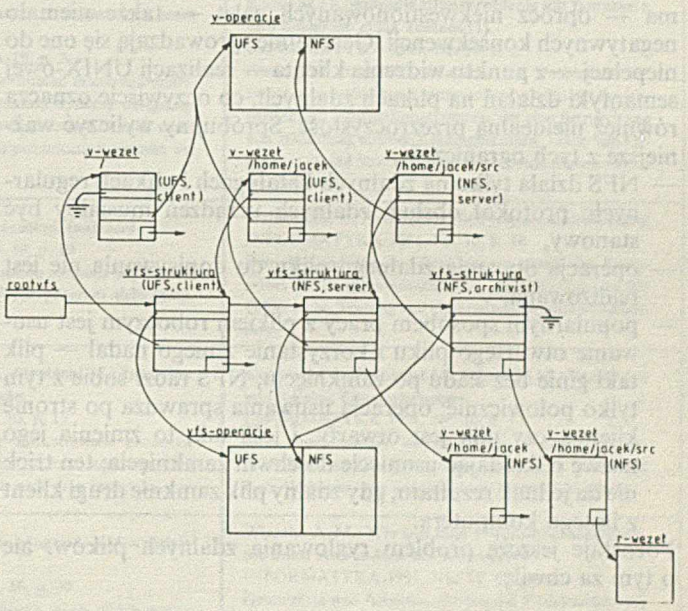
Rys. 3. Węzeł wirtualny i zdalny

Zacznijmy od v-węzłów. V-węzeł (rys. 3) stanowi swego rodzaju uogólnienie i-węzła, choć go nie zastępuje. Swój v-węzeł ma m.in. każdy otwarty plik, każdy punkt montowania oraz korzeń. Każdy v-węzeł zawiera wskaźnik do vfs-struktury swego macierzystego systemu plików oraz wskaźnik do analogicznej struktury systemu plików, dla którego jest punktem montowania (dzięki temu punktem montowania może stać się dowolny węzeł w drzewie katalogów). V-węzeł zwykle ma również wskaźnik do „realnego" węzła pliku: do i-węzła w przypadku pliku lokalnego lub do **r-węzła** (z uchwytem) w przypadku pliku zdalnego.



Rys. 4. Struktura danych zamontowanego zdalnego systemu plików

Swoją vfs-strukturę (rys. 4) ma każda zamontowana hierarchia plików (system plików lub poddrzewo). Poza operacjami odnoszącymi się do danej hierarchii jako całości vfs-struktura wskazuje m.in. na v-węzeł punktu montowania, co umożliwia trawersowanie ścieżek zawierających „.." również poprzez punkty montowań.



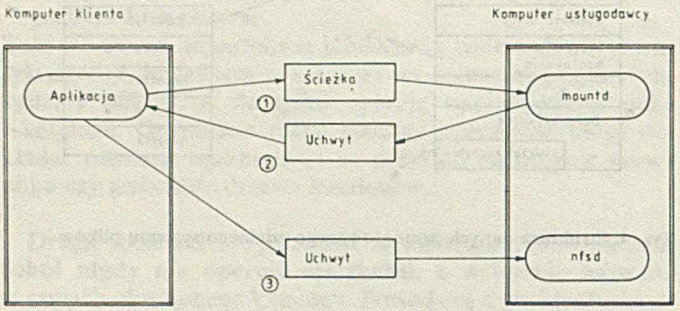
Rys. 5. Niektóre powiązania vfs-struktur i v-węzłów w hierarchii katalogów z rys. 1

Niektóre powiązania v-węzłów i vfs-struktur dla hierarchii plików z rys. 1. pokazuje rys. 5.

Ścieżki i montowanie

Wiemy już, że NFS-owy protokół pomija montowanie i nie operuje pojęciem ścieżki. Zauważmy jednak, że NFS nie mógłby istnieć bez operacji montowania — bo skąd klient wziąłby uchwyt korzenia zamontowanej hierarchii? W UNIX-owej

implementacji montowania aplikacja komunikuje się najpierw z demonem montowania **mountd**, posyłając mu ścieżkę do katalogu-korzenia zdalnej hierarchii plików, którą chce zamontować (rys. 6). W odpowiedzi demon zwraca uchwyt tego katalogu. Dopiero teraz klient jest w stanie nawiązać sensowną współpracę z demonem NFS-u. Warto zwrócić uwagę, że po stronie usługodawcy nie przechowuje się żadnego śladu dokonanych montowań — również protokół montowania jest bezstanowy.



Rys. 6. Współpraca aplikacji klienta ze zdalnymi demonami *mountd i nfsd*

Źródła kłopotów

Narzucone przez NFS-owy protokół założenie bezstanowości ma — oprócz niekwestionowanych zalet — także niemało negatywnych konsekwencji. Generalnie sprowadzają się one do niepełnej — z punktu widzenia klienta — realizacji UNIX-owej semantyki działań na plikach zdalnych, co oczywiście oznacza również nieidealną przezroczystość. Spróbujmy wyliczyć ważniejsze z tych ograniczeń;

— NFS działa tylko na zdalnych katalogach i plikach regularnych; protokół obsługi zdalnych urządzeń musiałby być stanowy,
— operacja otwarcia zdalnego pliku do dopisywania nie jest realizowana,
— popularnym sposobem pracy z plikiem roboczym jest usuwanie otwartego pliku i korzystanie z niego nadal — plik taki ginie bez śladu po zamknięciu; NFS radzi sobie z tym tylko połowicznie: operacja usuwania sprawdza po stronie klienta, czy plik jest otwarty, i jeśli tak, to zmienia jego nazwę odkładając usunięcie do chwili zamknięcia; ten trick nie da jednak rezultatu, gdy zdalny plik zamknie drugi klient z innego komputera.

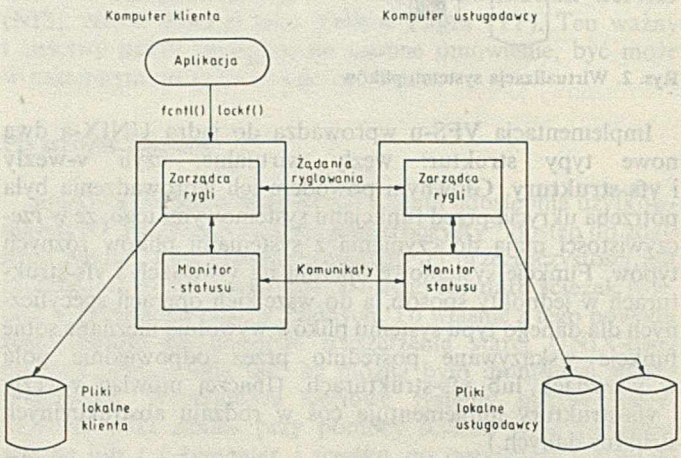
Pozostaje jeszcze problem ryglowania zdalnych plików, ale o tym za chwilę.

Natomiast zupełnie inne jest źródło pewnej uciążliwości nękającej w niektórych implementacjach NFS-u uprzywilejowanych użytkowników. Zwykle uważa się mianowicie, że uprawnienia lokalnego superużytkownika nie powinny rozciągać się bez ograniczeń również na pliki zdalne. W niektórych implementacjach usługodawca sprawdzając uprawnienia obsługiwanego nielokalnego superużytkownika przyjmuje jego uid za równy -2 (*nobody*), co czasami oznacza uprawnienia niższe niż przysługujące zwykłym użytkownikom.

Inna dosyć poważna niedogodność wynika z tego, że usługodawca realizując zdalne wywołania procedury **lookup( )** nigdy nie przekracza punktów montowań. Oznacza to, że jeśli zdalna hierarchia posiada domontowaną po stronie usługodawcy inną hierarchię, to klient nigdy jej nie zauważy, chyba że sam ją zamontuje. W przykładzie z rys. 1 użytkownik nie widziałby plików z komputera *archivist*, gdyby kolejność montowania była odwrotna.

Mechanizm ryglowania

Ryglowanie jest operacją z natury rzeczy stanową, i jako takie jest nie do pogodzenia NFS-owym protokołem. Problemy z ryglowaniem uwidoczniają się zwłaszcza podczas awarii: gdy ulegnie jej komputer usługodawcy, klient będzie musiał w przyszłości odtworzyć swe rygle, jeśli będzie to komputer klienta, usługodawca będzie musiał zdecydować, czy od razu zdjąć jego rygle, czy też trochę z tym poczekać.



Rys. 7. Ryglowanie

W ryglowaniu zdalnych plików zastępuje NFS **zarządca rygli** (ang. *lock manager*) [5], usługa sieciowa, na której barki jądro UNIX-a składa całą odpowiedzialność za realizację funkcji systemowych **lockf()** i **fcntl()** związanych z ryglowaniem zdalnych plików. Zarządca rygli z kolei polega na usługach **monitora statusu** (ang. *status monitor*), którego funkcja sprowadza się do informowania zainteresowanych o podnoszeniu się komputerów w sieci po awarii. Schemat ich współpracy jest następujący (rys. 7):

1. Każdy z pary zarządców rygli rejestruje w lokalnym monitorze statusu swoje zainteresowanie stanem zdalnego komputera;
2. Monitor statusu informuje lokalnego zarządcę rygli o tym, że zdalny komputer podniósł się po awarii;
3. Jeśli była to awaria komputera klienta, zarządca rygli usługodawcy zwalnia wszystkie rygle tego klienta (i wszystkich klientów innych działających na komputerze klienta);
4. Jeśli była to awaria komputera usługodawcy, nasz klient (i wszyscy inni klienci tego usługodawcy) próbuje odtworzyć swoje rygle, co oczywiście nie musi się udać.

pc-nfs

Na zakończenie trzeba choćby wspomnieć również o **PC-NFS** — bardzo popularnej implementacji DOS-owej. PC-NFS realizuje tylko stronę klienta, co jest zrozumiałe zważywszy, że DOS jest systemem jednoprogramowym.

dokończenie na s. 32

Programowanie w języku awk (2)

W poprzedniej części artykułu przedstawiono zastosowanie języka **awk** do implementacji prostej bazy danych obsługującej finanse domowe. Aktualną część poświęcimy prezentacji **awk** jako narzędzia do pisania kompilatorów prostych języków. **Awk** był pomyślany przez autorów głównie jako narzędzie do testowania programów **grep** i **sed** wykorzystujących wyrażenia regularne do wyszukiwania tekstów; wkrótce okazało się, że język wyśmienicie nadaje się również do pisania kompilatorów języków bezkontekstowych. Głównym atutem była duża zwartość programów kompilujących napisanych w tym języku.

Kompilator prostego języka bezkontekstowego

W większości języków wysokiego poziomu występują struktury nawiasowe takie jak instrukcja typu **if-then-else** czy symbole **(i)** występujące w wyrażeniach, wymagające dużo silniejszych technik rozbioru gramatycznego niż języki regularne. W tej części artykułu chcieliśmy przedstawić Czytelnikom przykład programu **kalkulator** obliczającego wartość wyrażenia arytmetycznego zapisanego w klasycznej, wrostowej (infixowej) notacji. Prezentowany przykład, nieznacznie tylko zmieniony, wzięty jest z autorskiej pracy [1].

Wyrażenie arytmetyczne, w którego skład wchodzą operatory dodawania, odejmowania, mnożenia i dzielenia oraz liczby i nawiasy, można opisać następującą gramatyką bezkontekstową:

```
wyrażenie --> składnik
              wyrażenie + składnik
              wyrażenie - składnik
składnik  --> czynnik
              składnik * czynnik
              składnik / czynnik
czynnik   --> liczba
              ( wyrażenie )
```

gdzie symbolami nieterminalnymi są **wyrażenie**, **składnik**, **czynnik** i **liczba**, a symbolami terminalnymi + − * / ( ) raz cyfry i kropka.

Przedstawiona gramatyka nie tylko definiuje składnię wyrażenia arytmetycznego, ale i odzwierciedla priorytety operatorów arytmetycznych.

dokończenie na s. 27

Kuraś M.: Problemy komunikacji z użytkownikami i wewnątrz zespołu (1) INFORMATYKA 1991, nr 10, s. 2 Pierwsza część charakterystyki problemów komunikowania się ludzi podczas projektowania systemów informacyjnych, zawierająca omówienie znaczenia tego rodzaju komunikacji oraz organizacji i strategii procesu komunikacji.	**Kuraś M.: Intercommunication problems with users and inside of a team (1)** INFORMATYKA 1991, No. 10, p. 2 First part of charakteristics of human intercommunication problems during information systems designing, which includes discussion on significance of such communication, as well as on communication process organization and strategy.	**Kuraś M.: Kommunikationsprobleme mit Benutzern und innerhalb eines Teames (1)** INFORMATYKA 1991, Nr. 10, S. 2 Erster Teil einer Charakteristik von Problemen der zwischenmenschlichen Kommunikation während Projektierung von Informationssystemen, der eine Besprechung von Bedeutung solcher Kommunikation, sowie von Organisation und Strategie eines Kommunikationsprozesses, umfasst.
Kalinowski T.: Analizatory wykonania programów INFORMATYKA 1991, nr 10, s. 10 Zasady działania oraz sposób korzystania z analizatora wykonania programu — narzędzia służącego do uruchamiania, testowania i optymalizacji programów.	**Kalinowski T.: Program execution analysers** INFORMATYKA 1991, No. 10, p. 10 Operation principles and utilization methods of program execution analyser — a tool for program debugging, testing and optimizing.	**Kalinowski T.: Analysator für Programmausführung** INFORMATYKA 1991, Nr. 10, S. 10 Betriebsgrundsätze und Benutzungsmethode des Analysators für Programmausführung, eines Werkzeuges für Fehlersuche, Testung und Optimalisation eines Programmes.
Kokowski T., Reformat M., Stroiński M.: Łączenie lokalnych sieci komputerowych INFORMATYKA 1991, nr 10, s.17 Charakterystyka produktów sprzętowych i programowych umożliwiających rozszerzenie sieci lokalnych i ich łączenie z sieciami rozległymi.	**Kokowski T., Reformat M., Stroiński M.: Interconnection of local area networks** INFORMATYKA 1991, no. 10, p. 17 Characteristics of hardware and software products, which makes possible extension of local area networks and their interconnection with wide area networks.	**Kokowski T., Reformat M., Stroiński M.: Verbinden von lokalen Computernetzen** INFORMATYKA 1991, Nr. 10, S. 17 Eine Charakteristik von Hardware - und Softwareprodukten, die eine Erweiterung der lokalen Netze und ihres Verbinden mit ausgedehnten Netzen ermöglicht.
Wierzba A. M.: Jak pisać programy satysfakcjonujące użytkownika? INFORMATYKA 1991, nr 10, s. 20 Zasady i metody pracy projektanta zapewniające uzyskanie programu bardzo dobrej jakości.	**Wierzba A. M.: How to write programs which satisfy its user?** INFORMATYKA 1991, No. 10, p. 20 Rules and methods for designer's work which secure a program of very good quality.	**Wierzba A. M.: Wie soll man Programme schreiben, die ihren Anwendern zufriedenstellen?** INFORMATYKA 1991, NR. 10, S. 20 Grundsätze und Arbeitsmethoden des Projektanten, die Erreichung eines Programmes mit sehr hohen Qualität garantieren.
Bronowska M.: Procedury graficzne systemu RAMTEK RM-9460 INFORMATYKA 1991, nr 10, s. 25 Charakterystyka rozwiązań pakietu programowego ułatwiającego programowanie grafiki w systemie RAMTEK.	**Bronowska M.: Graphic procedures of the RAMTEK RM-9460 system** INFORMATYKA 1991, No. 10, p. 25 Characteristics of solutions of program package which makes easy graphic programming in the RAMTEK system.	**Bronowska M.: Grafische Prozeduren des RAMTEK RM-9460 — Systems** INFORMATYKA 1991, Nr. 10, S. 25 Eine Charakteristik von Lösungen des Programmpakets, das Programmierung von Grafik im RAMTEK-System erleichtert.
Ostrowski P., Średniawa M.: Język specyfikacji SDL (2) INFORMATYKA 1991, nr 10, s. 28 Druga część charakterystyki języka SDL, zawierająca omówienie problemów zastosowań oraz tendencji rozwojowych.	**Ostrowski P., Średniawa M.: The SDL specification language (2)** INFORMATYKA 1991, No. 10, p. 28 Second part of the SDL language characteristics, which includes discussion of application problems and development trends.	**Ostrowski P., Średniawa M.: SDL-Spezifizierungssprache (2)** INFORMATYKA 1991, Nr. 10, S. 28 Zweiter Teil einer Charakteristik von der SDL-Sprache, der eine Besprechung von Anwendungsproblemen und Weiterentwicklungstendenzen umfasst.

Architektura PA-RISC w nowych modelach rodzin HP 3000 oraz HP 9000

W dniu 24 czerwca br., a więc po upływie niespełna trzech miesięcy od chwili wprowadzenia na światowy rynek komputerowy rewelacyjnych stacji roboczych HP Apollo 9000 seria 700 (patrz INFORMATYKA nr 8, 1991, s. 28), HEWLETT-PACKARD zaskoczył środowisko profesjonalne nie mniej rewelacyjną ofertą nowych produktów, które jeszcze bardziej umocnią pozycję tej firmy, jako czołowego prekursora postępu technologicznego.

Rewelacyjność oferty polega na wprowadzeniu do sprzedaży aż szesnastu nowych modeli, znacznie rozszerzających — dzięki zastosowaniu architektury PA-RISC — dotychczasowe możliwości oraz efektywność użycia w sferze zarządzania, bardzo popularnych rodzin komputerów HP 3000 oraz HP 9000. Przy tegorocznym tempie wprowadzania przez firmę nowych wyrobów, ostatnie strategiczne posunięcie HEWLETT-PACKARD zaskakuje bezprecedensową szerokością oferty, adresowanej do nowych kategorii użytkowników. Z dużym prawdopodobieństwem można więc spodziewać się, że marcowa i czerwcowa ofensywa najnowocześniejszej technologii, pokrywająca zapotrzebowanie wszystkich znaczących obszarów współczesych zastosowań profesjonalnych, w krótkim czasie doprowadzi do istotnych zmian kolejności pozycji w światowej czołówce producentów sprzętu komputerowego.

Znaczenie najnowszego osiągnięcia HEWLETT-PACKARD polega jednak nie tylko na niespotykanym dotąd rozmiarze jednorazowej oferty, ale w pierwszym rzędzie na tym, że w rodzinach komputerów ukierunkowanych na zastosowania związane z zarządzaniem wprowadzono architekturę RISC, spotykaną dotąd wyłącznie w obszarze zastosowań naukowych i technicznych, a także, że architekturę tę wprowadzono w systemach małych i średniej wielkości. Oznacza to, że HEWLETT-PACKARD oferuje obecnie najszerszy na świecie asortyment komputerów opartych na architekturze RISC, od komputerów biurkowych (ang. desktop) przez stacje robocze do dużych systemów, dostosowanych do wymagań scentralizowanego zarządzania wielkich przedsiębiorstw.

Architektura PA-RISC

Opracowana przez HEWLETT-PACKARD architektura PA-RISC (**Precision Architecture RISC**), otwierająca nowe możliwości wzrostu efektywności systemów komputerowych, znacznie obniżyła koszty projektowania tych systemów oraz skróciła czas pojawiania się nowych wyrobów na rynku, a także zwiększyła zainteresowanie ich zakupem. Według niezależnych analiz, udział w światowej sprzedaży wyrobów o architekturze PA-RISC w 1990 r. przekraczał 30% i co najmniej dwukrotnie przewyższał udział rynkowy każdego z następnych w kolejności czołowych producentów wyrobów o architekturze RISC.

Obecna przewaga HEWLETT-PACKARD jest wynikiem zaangażowania się firmy w technologię RISC już na początku lat osiemdziesiątych. Szybkie i odważne wprowadzenie tej technologii do rynkowych wyrobów, również w dziedzinie sprzętu do zarządzania, pozwoliło osiągnąć stosunek efektywności do ceny przewyższający tego rodzaju wskaźnik wykazywany przez firmy konkurencyjne. Szczególnie istotne jest wprowadzenie architektury PA-RISC do sprzętu ukierunkowanego na bieżące przetwarzanie transakcji (**On-Line Transaction Processing, OLTP**), stanowiące dominujące zastosowanie komputerów w przedsiębiorstwach, zwłaszcza we współczesnych rozwiązaniach sieciowych.

Do skokowego wzrostu efektywności systemów firmy HEWLETT-PACKARD przyczyniło się również wprowadzenie układów o bardzo dużym stopniu scalania (VLSI) opartych na najnowocześniejszej technologii CMOS. Pozwoliło to zaprojektować układy scalone o szczególnie dużej gęstości, zwiększającej szybkość i niezawodność działania procesora. Niezawodność ta, określana średnim czasem bezawaryjnego działania, kształtuje się obecnie w sprzęcie HEWLETT-PACKARD na niewiarygodnym poziomie ponad 43 000 godzin, tzn. ok. pięciu lat!

Nowe systemy rodziny HP 3000

Nowe systemy rodziny HP 3000 należą do serii 900 i obejmują 10 modeli o symbolach 917LX-967LX (6 modeli) oraz 937-967 (4 modele). Są to systemy od małej do średniej mocy obliczeniowej, dostarczane w obudowach przybiurkowych (ang. *deskside*) o bardzo zwartej konstrukcji — niektóre modele są tylko trochę większe od komputera osobistego! W wersji standardowej mają wbudowane od jednej do trzech stacji dysków nowej generacji, o bardzo dużej pojemności, oraz jedną stację jeszcze bardziej pojemnej pamięci kasetowej. Systemy są przeznaczone w pierwszym rzędzie dla małych i średniej wielkości przedsiębiorstw lub wydziałów dużych przedsiębiorstw i ich filii działających w ramach sieci. Stosowane mogą być na najbardziej newralgicznych stanowiskach pracy, wymagających bardzo szybkiego i niezawodnego przetwarzania. Oprócz jednoukładowego procesora, podstawowymi składnikami każdego modelu są:

- Pamięć wewnętrzna typu DRAM, zawarta w układach o pojemności po 1 lub 4 Mbitów oraz instalowana w modułach o pojemności 8-, 16- lub 32 Mbajtów. Ma wspomaganie bateryjne, zapewniające zachowanie informacji co najmniej 15 minut w przypadku przerwy zasilania z sieci. W zależności od typu modelu, pamięć ta może mieć pojemność od 24 do 192 Mbajtów.

- Pamięć typu cache, podzielona w równych częściach dla instrukcji oraz danych, w zależności od typu modelu może mieć pojemność od 96 do 512 Kbajtów.

- Pamięć dyskowa standardu 5,25 cala, która w zależności od typu modelu może obejmować w obudowie od jednej do trzech jednostek dyskowych o pojemności 673 lub 1355 Mbajtów. Łącznie z jednostkami zewnętrznymi pojemność tej pamięci może wynosić od 24 od 66 Gbajtów. Oprócz wielkiej pojemności, pamięć ta charakteryzuje się bardzo dużą niezawodnością działania — średni czas bezawaryjny wynosi powyżej 150 000 godzin!

- Pamięć kasetowa typu DDS (**Digital Data Storage**) umożliwia archiwowanie danych na taśmie magnetycznej kaset standardowych (73x54x10,5 mm) do cyfrowego zapisu dźwięku typu DAT (**Digital Audio Tape**). Taśma z zapisem 120-minutowym (60 m) ma pojemność 1,3 Gbajta. Za pomocą programu kompresji danych HP Turbo-STORE/XL II pojemność takiej kasety można zwiększyć do 5 Gbajtów.

- Gniazda rozbudowy wejścia-wyjścia — dwa w modelach 917LX -967LX, umożliwiające przyłączenie do 18 stacji dysków i do 6 stacji taśm kasetowych DDS, lub dwanaście w modelach 937-967, umożliwiających przyłączenie do 49 stacji dysków i do 8 stacji taśm kasetowych DDS.

- 64-bitowy koprocesor operacji zmiennoprzeciwnych — dostarczany na żądanie tylko w modelach 957LX i 967LX oraz 957 i 967.

- Obudowa, która w modelach 917LX-967LX ma wysokość 430 mm, szerokość 222 mm i głębokość 533 mm oraz masę 31,8 kg, a w modelach 937-967 przy identycznej wysokości i głębokości — podwójną szerokość (444 mm) oraz masę 50 kg.

Z systemu może jednocześnie korzystać w przypadku modeli LX od 8 do 600 użytkowników, a w przypadku modeli 937-967 — od 32 do 600.

Orientacyjne ceny standardowych konfiguracji są obecnie na rynku USA następujące: modele LX od ok. 25 000 do ok. 284 000 dolarów, natomiast modele 937-967 od ok. 90 000 do ok. 367 000 dolarów. Ceny konfiguracji standardowej obejmują system operacyjny MPE/XL. Zgodność z definicją systemu otwartego X/Open umożliwia bezkonfliktowe włączanie nowych modeli do najbardziej popularnych systemów sieciowych oraz łatwe przenoszenie oprogramowania.

Nowe systemy rodziny HP 9000

Nowe systemy rodziny HP 9000 należą do serii 800 i obejmują 6 modeli o symbolach 807S — 857S. Ze względu na ich główne przeznaczenie, jakim jest funkcja serwerów sieciowych, są one określane nazwą serwerów przedsiębiorstwa (**Business Servers**). Podobnie jak nowe modele rodziny HP 3000, są to systemy od małej do średniej mocy obliczeniowej, dostarczane w dwóch wariantach obudowy o identycznych gabarytach oraz wyglądzie zewnętrznym (obudowa mniejsza dla modeli 807S, 817S i 837S, obudowa większa dla pozostałych modeli). Mimo niewielkich gabarytów, niewiele przekraczających wymiary komputera osobistego, efektywność cenowa, zwłaszcza modeli najmniejszych, jest wielokrotnie lepsza od efektywności podobnych systemów firm IBM oraz DEC, a w przypadku modeli większych ich moc obliczeniowa jest porównywalna z dużymi systemami minikomputerowymi tych firm.

Wszystkie podstawowe składniki konstrukcji nowych modeli serii 800 są identyczne z występującymi w nowych modelach rodziny HP 3000, z tą jednak różnicą, że dolna (a w niektórych przypadkach również górna) granica łącznej pojemności poszczególnych rodzajów pamięci (z wyjątkiem pamięci typu cache) jest niższa i wynosi:

- dla pamięci wewnętrznej — od 8 do 192 Mbajtów,

- dla pamięci dyskowej — od 0,23 do 47,6 Gbajtów, przy czym oprócz standardu 5,25 cala są stosowane również dyski standardu 3,5 cala,

- dla pamięcie taśmowej DDS — od 1,3 do 5,2 GBajta.

Koprocesor operacji zmiennoprzecinkowych jest niedostępny tylko w modelu 807S, natomiast w pozostałych modelach jest on instalowany na żądanie lub stanowi obowiązkowy składnik konfiguracji standardowej (modele 847S i 857S). Podobnie jak w nowych systemach rodziny HP 3000, poszczególne modele serii 800 mogą mieć od 2 do 12 gniazd rozbudowy wejścia-wyjścia, a w zależności od typu modelu, z systemu może korzystać jednocześnie od 16 do 160 użytkowników.

Orientacyjne ceny standardowych konfiguracji poszczególnych modeli serii 800 są niższe niż serii 900 i wynoszą obecnie na rynku USA od ok. 17 000 do ok. 127 000 dolarów. Ceny konfiguracji standardowej obejmują system operacyjny HP-UX, zgodny z systemem UNIX oraz specyfikacjami X/Open XPG3, POSIX, 1003.1 oraz SVID2. Oznacza to, że nowe modele rodziny HP 9000 spełniają warunki systemów otwartych, a więc mogą bezpośrednio współpracować ze sprzętem komputerowym innych producentów. Umożliwia to korzystanie z wielu tysięcy programów użytkowych, w tym również renomowanych produktów czołowych wytwórców oprogramowania.

Po szczegółowe informacje prosimy zwracać się do:

Hewlett-Packard Polska Sp. z o.o.
ul. Newelska 6, 01-447 Warszawa
tel. (22) 36-83-00, faks (22) 37-47-83, teleks 813919

0/13/91

światowy lider architektury RISC

UMC
UNITED MICROELEKTRONICS CORPORATION
ZNANY PRODUCENT UKŁADÓW SCALONYCH PROPONUJE:

* UKŁADY PAMIĘCI
  SRAM, od 4K do 1 Mb
  MaskROM, od 64K do 8Mb
* UKŁADY KOMPUTEROWE
  Kontrolery peryferyjne UM8250..., um82450...
  Komplety chipsów do AT, 386, 486
  Układy Arcnet/fax/Modem
* UKŁADY KOMUNIKACYJNE I KOMERCYJNE
  Układy telefoniczne zwykłe i inteligentne
  Układy zdalnego sterowania (TV itp.)
  Układy do systemów alarmowych
  Generatory melodii UM66..., UM 348...
  Układy do zastosowań specjalnych

OFICJALNY PRZEDSTAWICIEL:
meditronik SPÓŁKA z o.o.
00-194 Warszawa, ul. Dzika 4
tel. (02) 635 22 63, 635 22 64
fax (02) 635 21 95, tlx 816 075

0/24/90

Micomp®

Integracja systemów ICL (1900, ME 29, 2900, 39) i IBM z mikrokomputerami PC i systemami NOVELL, UNIX. Sieci lokalne (LAN) i rozległe (WAN).

- Produkty własne MiCOMP i czołowych producentów światowych:
  - * MiCOMP - 8075 - kontroler grona,
  - * MiCROSS - FXBM - emulacja ICL 7503+/7561,
  - * MiCROSS - NVL - most ICL - NOVELL,
  - * MiHQ - LAN - most ICL, IBM, DEC - NOVELL, UNIX,
  - * Bisync, X - 25, OSLAN, TCP/IP, ICL CO3, SNA
- Instalacje sieci w oparciu o technologie BICC.
- Kompleksowa obsługa klienta: od projektu do uruchomienia systemu i szkolenia .
- Instalacje w największych systemach w kraju,
- 7 lat działalności na rynku.

Wielodostępne systemy SCO UNIX/OPEN DESKTOP

- komputery ALR (PowerPro, MultiAccess),
- oprogramowanie: SCO UNIX , SCO OPEN DESKTOP, INFORMIX,
- integracja sieci- technologia BICC,
- adaptery inteligentne i proste , modemy, konwertery, terminale,
- projektowanie, konsultacje, szkolenie, "gorąca linia".

DYSTRYBUCJA

Monitory ekranowe MICOMP monochromatyczne i kolorowe

- wysoka jakość obrazu
- niski poziom radiacji
- pełny serwis gwarancyjny

MiCOMP MAGISTRAT '91 Kompleksowy System Komputeryzacji Urzędu Gminy

- standard stosowany przez większość urzędów na Śląsku,
- nowoczesna konstrukcja oprogramowania,
- integracja w systemie NOVELL,
- kompleksowa obsługa klienta: oprogramowanie, sprzęt, instalacja, szkolenie, wdrażanie i pielęgnacja.

MiCOMP
Dział Handlowy
ul. Astrów 7 , 40-045 KATOWICE
TEL: 513086 FAX: 518628 TLX: 315687

0/11/91